1989

Amused: a multi-user software environment diagnostic

Mary Ann Foltman

Rochester Institute of Technology
School of Computer Science and Technology

AMUSED:
A
Multi-User
Software Environment
Diagnostic

by
Mary Ann Foltman

A thesis, submitted to
The Faculty of the School of Computer Science and Technology,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Approved by:

_____
John A. Biles

_____
Karen K. Anderson

_____
Peter G. Anderson

17 May 1989

THESIS RELEASE PERMISSION FORM

ROCHESTER INSTITUTE OF TECHNOLOGY
College of Applied Science and Technology

Title of Thesis:
AMUSED: A Multi-User Software Environment Diagnostic

I, Mary Ann Foltman, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

_____
Mary Ann Foltman

Abstract

As software projects grow in size and complexity, many individuals take over the responsibilities for one project, creating a potential for new errors in the development process. Software version inconsistency, unfamiliarity with the tools used, and software tool restrictions are but some of the problems encountered in a multi-programmer environment. These problems are not always self-evident to the programmer and may require a dedicated software support representative or experienced programmers to assist. These problems can be reduced through the development of a multi-user software environment diagnostic expert system, AMUSED (A Multi-User Software Environment Diagnostic). The AMUSED expert system is designed for use by programmers responsible for creating the executable software releases on a standard copier / duplicator project. Project source code is transported to a common workstation, and linked together with other programmers' code through a linking tool. AMUSED's diagnostic help assists (a) the link process that will be used to create the executable code from the source files, (b) the retrieval of source files from remote sites to the link workstation, and (c) the use of any interfacing connections between the source modules.

Keywords

Expert system, software configuration management, software development, multi-user environment, diagnostics.

Acknowledgements

During the entire thesis process, a network of people was established to help support this project. Numerous people provided resources and references during the course of development. Among these people are my advisors, who include: Al Biles, Rochester Institute of Technology graduate instructor; Karen Anderson, a systems engineer at Xerox Corporation; and Dr. Peter Anderson, Graduate Computer Science Department Head at RIT.

For the development phase of the project, several people who were familiar with the tools used were available for consultation. Dennis Brantly assisted with the use of Interlisp, and Kathy Matysek assisted with problems with Common Lisp. People on various software projects at Xerox and Rank Xerox were also contacted. These contacts provoded useful information on how errors are handled in each group, as well as on which errors are most common and critical. They also supplied much of the data which was entered into AMUSED. The contacts included: Mike Sprague, Mary Beth Anderson, Karen Maloney, Bob Harwood, and Marlene Yaw of Xerox Corporation in the USA, and Dennis Miazga and Dave Bradley of Rank Xerox at Welwyn Garden City, England. Other people within Xerox have been able to provide other information as well. This information was often in the form of reference articles, new people to contact, or other areas to research. Among these people are Bill Anderson, Jack Bacon, and Cynthia Nyborg.

Other people that I wish to acknowledge include my managers at Xerox, who supported my schooling over the past four years, Bill Kukucka, Jim Petery, and Jerry Boesl.

AMUSED Table of Contents

1. Introduction 1
    1.1 The Multi-User Software Environment 1
    1.2 The General Software Development Process 2
    1.3 An Expert System Application 3
    1.4 Chapter Summaries 4
2. Background 5
    2.1 Review of Past Projects 5
        2.1.1 Software development projects 5
        2.1.2 Related Expert Systems 6
        2.1.3 Related Xerox Projects 20
    2.2 Languages and Tools 26
        2.2.1 Common Lisp 26
        2.2.2 TMYCIN / EMYCIN 28
            2.2.2.1 EMYCIN 28
            2.2.2.2 TMYCIN 30
        2.2.3 Discarded Tools 31
    2.3 Present System at Xerox 35
3. Implementation 38
    3.1 System Overview 38
    3.2 AMUSED Appearances 39
    3.3 User Tutorial 42
        3.3.1 Diagnose Walkthrough 42
        3.3.2 Assist Walkthrough 47
    3.4 Technical Overview 50
        3.4.1 Basic Control Flow 50
        3.4.2 Diagnose Options 51
        3.4.3 Initial Questions 52
        3.4.4 Asking Questions 53
        3.4.5 Processing Questions 55
        3.4.6 Certainty Factors 56
        3.4.7 Distributed Rule Bases 56
        3.4.8 Question Database 58
        3.4.9 Assist Mode 60
    3.5 Summary 60
4 Testing 64
    4.1 Data Collection 64
    4.2 Early Implementation Results 65
    4.3 Testing 66
5. Conclusion 67
    5.1 AMUSED Shortcomings 67
    5.2 Alternate Approaches 68
    5.3 Possible Extensions 69
AMUSED References 71
Appendix A -- DF File Sample 74
Appendix B -- TMYCIN Data Flow Diagrams 76
Appendix C -- TMYCIN Function Description 81
Appendix D -- AMUSED Flow Charts 82
Appendix E -- AMUSED Source Code 86

1. Introduction

1.1 The Multi-User Software Environment

Over the years, software projects have grown in size and complexity. Many projects have expanded to such magnitudes that they cannot be written and maintained by one or even a few programmers. When more than one programmer takes over the responsibility for a given software module or subsystem, a layer of complexity is added to the original project. This complexity is compounded by the challenges of maintaining software consistency.

Programs that consist of a large number of modules need to be managed. When the number of modules making up a system exceeds some small, manageable set, a programmer cannot be sure that every new version of each module in the program will be handled correctly. After each version is created, it must be compiled or assembled. The programmer then may need to store it somewhere so others may use it. The versions of the software being transferred cannot be guaranteed to be correct without some form of assistance [SCHMIDT82].

During development and maintenance of these large software systems, a difficult problem always has been ensuring that the correct versions of source and object files have been used. As the number of programmers and software modules increases, the cost of mistakes escalates. This problem also expands when software references are duplicated across a network of personal workstations. It is complicated further if the software is developed in a strongly typed programming language such as Xerox' Mesa, which demands that all references to a module agree exactly in software version. Such version checking provides another check in the software process and allows errors to be caught sooner in the software development cycle. Such additional work up front, however, also can create frustration in the push to reach deadlines [LEWIS].

1.2 The General Software Development Process

Software development often involves a common set of steps specific to a given environment. Whether the software is developed in a high level language or in assembly code, a certain set of steps needs to be followed to produce the desired result. Many of these steps are dependent upon the environment being used and the size of the project. The use of software tools generated within Xerox for that specific software development environment helps to guarantee process commonality.

In the multi-user environment, programmers individually write their own software, but the amount of individual testing that can be done is limited. The software modules are next assembled or compiled. Modules are then stored in a common facility such as a file server or workstation for later retrieval. As in some high level languages, two separate sections of a module exist to register the interdependencies of the modules. The *interface* modules define the data abstractions and the *implementation* modules implement and use the data listed in the interfaces. When this format is used, the necessary completed interface modules are retrieved as needed. The first two steps of the process, that of compilation/assembly and storage of modules, are then performed on the implementation modules.

Once all code changes have been completed, all modules are retrieved onto a designated workstation, where they are combined through another software tool, such as the Xerox tools Link and Link8051, into an executable version of the software. Link and Link8051 are absolute mapping tools that take source code from varying languages and map them directly to a resulting executable file with the correct addresses needed by the specified Intel processor. This executable file could be a memory image file (MIF file) which is specifically used as input to the software tools for interactive testing, an Intel object format file, or a hex file. The hex files, most commonly used in many Xerox products, are usually generated for a variety of Intel Corporation processors, including Intel 8085, Intel 8051, and Intel 8086. An Intel hex file is created from the MIF file once it is loaded into the test tools. The object format file can be used in interactive testing, but is rarely used at Xerox. The hex file is the

input used to create the EPROMs to run the products, but it also can be used as an interactive test file.

Several common activities arise from the linking process. The most common activity is integrating the assorted software modules into an executable file using the Link and Link8051 tools. These two tools are needed to combine a wide variety of languages into a common executable element. The file manipulation scheme and the linking tools are the main areas of interest addressed by this project.

One of these areas of interest is the manipulation of files among locations. Three main tools are commonly used for the storage and retrieval of files. The File Transfer Program (FTP) is a command set run from the executive window in the Xerox Development Environment (XDE). It is most commonly used in association with a file that lists the commands to be executed with the necessary switches. FileTool is an independent tool that provides a better interface to the user, but it performs the same basic functions as FTP. The Describe File Tool (DFTool) uses a description file, known as a DF file, to allow files to be stored and retrieved while monitoring the versions being used. This tool will be discussed in depth later in the paper.

1.3 An Expert System Application

A variety of software problems can occur on a copier/duplicator project, especially when the project incorporates multiple modules, versions, and storage locations. With the proper precautions, many integration problems and other errors can be caught and fixed before the code reaches the hardware to be tested. AMUSED is an expert system tool that will provide programmers in a multi-user environment with diagnostic assistance in the linking or absolute mapping phase of a software project. AMUSED makes it easier to pinpoint problems with the basic linking package, as well as problems with file versions, and variable misuse. By circumventing the software preparation stage, AMUSED increases its potential usability, since many software projects at Xerox use one of the two main linking tools with a variety of software languages.

1.4 Chapter Summaries

This paper describes the AMUSED system in depth and provides an informational background for related topics. Chapter One has been an introduction to the process requiring the tool. Chapter Two will provide background on other research and projects in the same and related fields, as well as in-depth information into the tools used for the project. Chapter Three explains the actual implementation of the project. Chapter Four examines the results of the implementation including user sampling and response times. Chapter Five contains the conclusions obtained from the research. Appendix A contains a sample of DF file. The TMYCIN data flow diagrams are in Appendix B, while Appendix C contains a further description of the funcitons within TMYCIN. Appendix D contains the flowcharts describing the layout of AMUSED, and the final appendix contains a copy of the source code for AMUSED.

2. Background

2.1 Review of Past Projects

While much development has happened in recent years in the area of version control within Xerox, none of the applications has incorporated expert systems or artificial intelligence. In contrast, a small number of systems outside Xerox have been developed that apply some degree of artificial intelligence to the software development process, but none execute proper version control. Although only a small number of related references were found, those resources did offer good insight to past works, including those with artificial intelligence applications and those without.

2.1.1 Software development projects

Although expert systems and the use of artificial intelligence are the major concerns with the AMUSED project, some interesting insights were found from a project in which a software maintenance scheme was developed for a large European computer manufacturer. Similarities were found between large software projects and machinery with assemblies, sub-assemblies and parts [MARC84]. This connection also brought up the thought that good design naturally leads to maintainable software, but maintainable software requires more than a good software design.

Controlling software manufacturing requires a centralized database to collect information on the software such as status, quality, and structure. This database concept works like an expert system to handle the system complexities. Differences can be noted between well-developed software, which needs a one-time controlled configuration process, and well-maintained software, which can be regenerated at any time [MARC84]. Information on configuring a particular version of a product also must be preserved to allow that configuration to be reproduced reliably in the future.

One method for controlling the reconstruction of software configurations is to enforce change control. Use of change control regulates updates to a set of software by forcing the changes to be registered and reviewed by a select committee before being implemented. This control method helps users to

ensure that compilations and tests of components are done identically for each iteration. Correct versions of all tools used must also be tracked to ensure the proper configuration.

2.1.2 Related Expert Systems

While advances have been made in expert system development and expecially in integrated environments, Franco Manucci [MANU84] mentions that only two basic streams of research have evolved toward improving the realm of integrated environments, the evolutionary and revolutionary approaches. The evolutionary approach is conservative, aimed at incremental improvements, while the revolutionary approach is geared toward replacing the system with an automation-based paradigm using artificial intelligence techniques. Manucci goes on to state that these two approaches will develop in parallel for a time, and that there may be some applications of artificial intelligence that will tackle specific sections of the problem. However, he doesn't feel that artificial intelligence could be used for an entire comprehensive system at the present time [MANU84].

In contrast, Boyd [BOYD84] feels that software technology is a good candidate for the application of knowledge-based expert systems. Says Boyd, "Expert systems are distinguished from other conventional computer systems by their explicit use of a stored knowledge base manipulated by a clearly identifiable control strategy (the inference engine)" [BOYD84]. Such systems are capable of achieving a level of performance comparable to a human expert in a specialized domain. Boyd states that this domain could be in one of three general areas: the application domain, the execution environment, or the programming methodology. AMUSED falls into the programming methodology category, since it deals with the software development process.

Interactive Program Design Expert System

In discussing interactive program design, Harandi [HARA83] presents a new implementation method using knowledge-based techniques. Software design is often delegated to experienced software engineers who rely on past knowledge of similar products to design new systems / programs. However, by combining a knowledge-base of abstract design components, design heuristics, and various other related information with computer inference, a design can

be constructed from a user specification. This system uses a knowledge base with dataflow model segments to represent program design components. These segments are combined and refined, using transformation rules, to produce a dataflow model of the goal program.

A dataflow context diagram, while natural for humans to use, is also capable of representing a program model accurately and in an understandable format. The types of data that flow through a program and the transformations that process those dataflows are the terms that specify the program. The interior structure of any transformation may be detailed further by describing a data flow diagram (DFD), which specifies how the function of the transformation is accomplished in terms of lower level functions [HARA83]. When these data flow diagrams, together with a dictionary of data definitions, are collected, they can be used to adequately specify many types of programs.

The design aid system implements a set of specifications into a set of schemas which the user then can analyze for design intent. Several major units compose the system: the knowledge base, consisting of schematic system design information; a data dictionary; and knowledge about various domains of application. The design refinement unit is the inference engine of the system, controlling the design process. Responsible for all the user-system interactions is the user interface. The organization of the tool can be seen in Figure 2.1. Interactions with the tool occur through a restricted natural language interface, which is implemented through an augmented transition network (ATN) [HARA83]. A graphics interface also is provided to facilitate viewing and editing of the data flow design.

Both major components of the knowledge base, the catalog of schematic design components and the data dictionary, are arranged in an abstraction hierarchy to permit inheritance and sharing of common components. The meanings of the various components are domain oriented and are represented at different levels of abstraction.

The design aid tool functions by receiving specifications, scheduling goals for those specifications, and applying all refinement rules that can lead to

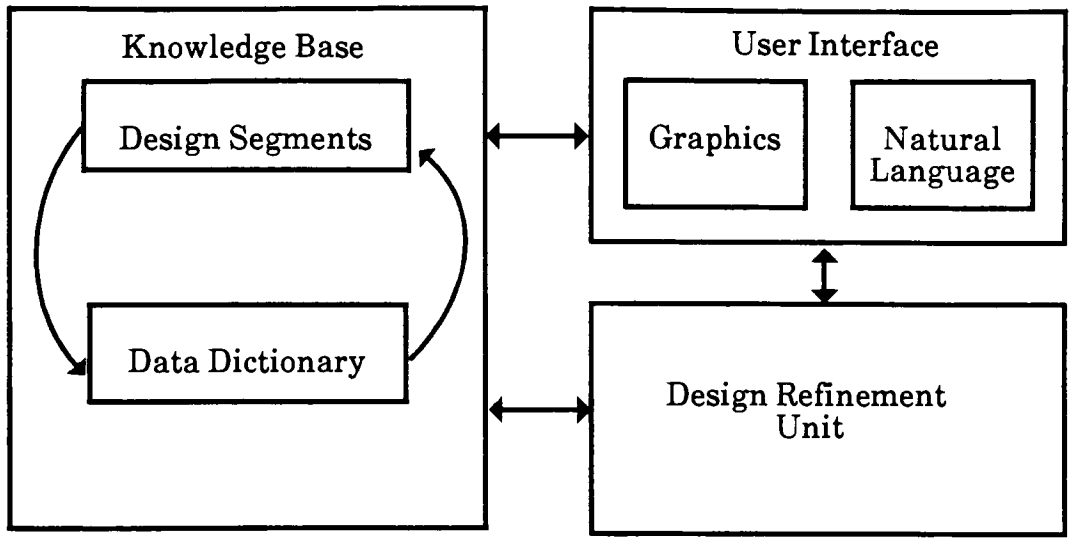


Figure 2.1. The Building Blocks of the Design Aid System

more detailed system designs. A list of goals is maintained and executed to manage the complexity of its various design decisions. The user is permitted to inspect and modify the design throughout the design process. Each modification is checked for consistency within the design model.

A final dataflow design can be used in a number of different ways. It can be transformed into a structured design using techniques such as transform and transaction analysis, simulated as a dataflow implementation or prototype, or used as input into a program editing tool. Users can use the tool as a reference to see particular types of dataflow models. It also can be used as a dataflow management tool.

The knowledge base is composed of a number of data dictionaries, which are used to understand and validate user requirements and specifications. Data definitions are contained in the data dictionary for all known data objects in the system's world model. The definitions also include such information as assumed values, defaults, constraints, properties, and predicates. Schemas for design components are stored in the data flow transformation dictionary. Information associated with these schemas

includes transformation definitions and submodels describing the input and output data flows, as well as references into the data dictionary.

An initial user specification is matched against the schemas for any related diagrams that already exist. This matching is done by names of objects, key words or phrases, or by exact words. If more than one possible response is found, the resolution is made by the program whenever possible or else, a choice is presented to the user. A list of goals to be addressed is maintained during the development process and is treated as a priority scheduling queue [HARA83]. Goals can be added to the front or the end of the queue, depending on urgency. The system addresses the next waiting goal on the highest priority queue until all queues are empty. Refinements of goals also are addressed on this queue.

The dataflow design aid tool is expected to help reduce error prone activities such as requirement analysis and specification. This help should be accomplished by allowing the requirements and specifications to be expressed at a level as close as possible to the conceptual level of the analyst. It would reduce the specification effort through the inclusion of domain knowledge in the tool. It also would perform validation of users specifications in conjunction with refinements to a dataflow design model.

Error Localization Expert System

Error localization in program debugging is the process of identifying program statements that cause incorrect behavior. A prototype of the error localization expert system for debugging Pascal programs was developed at Oakland University [KORE86]. The overlying goal in developing this system was to minimize the amount of information that the programmer needs to supply to locate the error. It makes use of the knowledge of program structure rather than the knowledge at the level of symptom fault rules. The knowledge of program structure is represented by a dependence network.

The process of debugging has generated comparatively little research, literature or formal instruction compared to other software-development activities. Some researchers say the error-locating process represents 95% of the debugging [KORE86]. Some of the reasons why the process is so difficult

include: the programmer must simultaneously keep track of inordinate amounts of detail; debugging a program requires a high degree of precision to isolate all aspects of an error and to track its total effect throughout the program; and a programmer's mind can become fixed on one possible cause even if the programmer is looking in the wrong place. Generally, the error-locating process is performed by means of break-and-examine debuggers. Through the use of breakpoint facilities a programmer inserts breakpoints around suspect instructions where the error is believed to originate and the program is re-executed until the offending instruction is found.

The Oakland prototype [KORE86] guides a programmer during the testing of Pascal programs. As an interactive system, it queries the programmer for the correctness of the program behavior and uses answers to focus the programmer's attention on an erroneous part of the program. The system makes use of knowledge of program structure rather than knowledge of symptom-fault rules. This deep-reasoning approach is different from the traditional shallow knowledge engineering approach (i.e. MYCIN), which captures diagnostic knowledge at the level of symptom-fault rules. The traditional approach gathers information over an extended period of time and becomes very program specific, which can be a disadvantage.

Program structure knowledge is represented by a dependence network, which is based on the concept of dependence relationship between program instructions. It is used by an error-locating reasoning mechanism to guide the construction, evaluation, and modification of hypotheses of possible causes of the error. Three different types of dependences can be found between program instructions in the execution trace. First is data influence, which is represented by data flow. Next is control influence, which is defined between the test instruction and the instructions that the test instruction can choose to execute or not execute. The last type is a control-data influence, which has both direct data and direct control influence on any instruction in the execution trace [KORE86].

Diagnostic problem solving is inherently sequential in nature. It is guided by a hypothesis-and-test process, where a programmer starts at the point the error is first noticed. The programmer next tries to reconstruct the sequence

of events leading up to the error. Generation of a hypothesis based on these conditions can be a problem, since it might not be the correct hypothesis or the correct position where the break occurred.

Korel's system, developed to be an error-locating assistant has five major components, as shown in Fig. 2.2 [KORE86]. A static analysis of the source

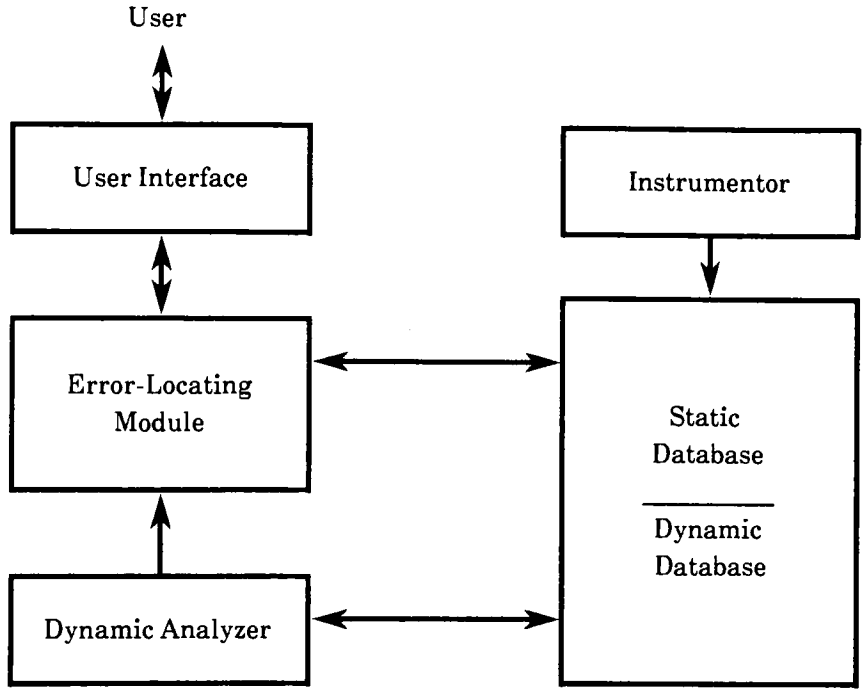


Figure 2.2 Organization of Error-locating Assistant

program text is performed by the Instrumentor tool. It then generates a static database and inserts the execution history, collecting instructions before every monitoring point in the source program, which creates an instrumented version of the original program. The database contains static information about the program and dynamic information about the program's execution. A static database is created automatically by the Instrumentor and consists of the following components: symbol table, instruction table, and program control flow graph. A dynamic database is created automatically during execution of an instrumented program. The dynamic database is a collection of program execution history records, one for each execution of the program.

The main function of the error-locating module is to reason about incorrect behavior of the program to guide the programmer during the localization process. This task is performed by creating the dependence network, generating the hypotheses of possible sources of error, based on the known findings of incorrectness, and evaluating these hypotheses based on querying the programmer about the correct/incorrect behavior of the program. The main function of the user interface is to provide communication between the programmer and the error-locating module. Automatically detecting the possible correctness in the execution trace is the purpose of the dynamic analyzer. The dynamic analyzer performs two types of analysis, uninitialized variable analysis and assertion analysis.

Ericsson Public Telecom System

A Telecom system is a very large program, but it is also one that exists in a number of different versions. Managing such a system is a difficult task, and the concepts offered in the field of artificial intelligence were studied to fill this need. It was clear that many of the problems when developing and managing big software systems concerns getting relevant information at the right time with a minimum of effort. "Object-oriented AI tools, also called frame systems, offer a method for organizing this information going beyond the capabilities of traditional databases" [WELI84].

In a frame oriented database, the information is organized as properties of objects corresponding to units of the application, for example, telecom software modules. Inheritance of properties and a hierarchy of objects could be created, reducing redundancy and allowing storage and display of information at different levels. This method would also allow consistency to be maintained.

Forward chaining production rules can be used either alone or in combination with an object-oriented database to make inferences about the software system and the design process [WELI84]. By checking the situations of a given design rule, the programmer can be warned if a rule were about to be violated. Constraint propagation and belief revision can be used to experiment with alternate solutions without changing the actual parameters. Such an ability would allow the effects of these systems' decisions to be

studied. Further support also could be provided were the proposed expert system integrated with the rest of the environment.

Ericsson Public Telecom Division has been conducting these experiments in artificial intelligence over the past few years. They decided it was better to acquire a basic technical competence in the field and to import and develop various kinds of artificial intelligence software, rather than to buy specialized expert system tools and start building expert systems at once [WELI84].

POZZO System

The knowledge-based approach to software development involves formulating the domain knowledge in a declarative way and leaving the procedural control to an inference engine [NORD86]. An important problem with this method of developing software is that it is difficult to express and represents domain-dependent control knowledge (i.e. how the object-level knowledge is organized by the person using it).

In studies of human expertise, it was discovered that an expert works by reusing solutions to previous problems [NORD86]. The use of prototypes, actual implementations of an expert's previous experience taken from typical cases to express and represent domain-dependent control knowledge, follows this example of reusing problem solutions. One important advantage of using prototypes as a method of representing control is that they are additive. The system can be run without them, but clarity and efficiency are gained from the additional information presented by the prototype cases.

POZZO was developed using prototypes in conjunction with a rule-based system. The prototypes are represented separately from both the domain knowledge and the inference engine, as illustrated in Figure 2.3. Prototypes found in POZZO contained several items: the parameter constraints, the range of parameter values, defaults and expressions for constraints; a list of values to be filled for a complete solution; knowledge about how to compose the advice once all values are obtained; strategic knowledge to guide the consultation and to explain to the user why things are done in a certain order; and pointers to instances of the prototype, saved for explanation and teaching [NORD86].

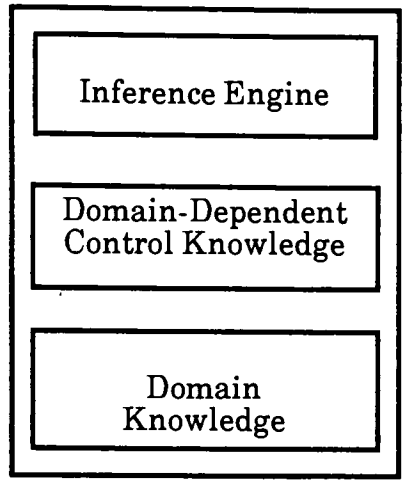


Figure 2.3. Model of knowledge-based system

These prototypes really have two types of knowledge: knowledge about when the prototype is applicable, and strategic knowledge of how to pursue a case once the prototype is invoked. Having prototypes guide the consultation offers several advantages. The system appears to behave in a more natural way, since the system has different strategies for different kinds of cases, and because these strategies can be explained to the user. Prototypes can be added to the system one at a time, providing the availability of a more concise answer.

POZZO combines the use of prototypes with a backward-chaining system consisting of rules and parameters. The prototypes set up the goals that the backward-chaining machine works on. A general prototype begins with general questions. When a specific prototype is invoked, that prototype takes the initiative and starts delivering goals in the order best suited for the case. If the prototype turns out to be incorrect, it is explained to the user, and the prototype is withdrawn.

A major problem with this approach, if it were used for production, would be knowledge acquisition. The expert needs to identify typical cases, and

these cases need to be implemented as prototypes. Also, if the rules and parameters were changed, what prototypes would then be valid [NORD86] ? The use of prototypes also could be used for the teaching of strategies and to generate interesting cases for the student.

Woodpecker System

The Woodpecker system is concerned with manipulating existing programs, which can be very costly when done by hand [FOUE84]. These manipulations can be needed either to modify a program slightly or to find and correct an error. An introspective look at the error-locating process begins with getting acquainted with the program. Next, questions are asked to determine what the problem is. The search space within the program is then reduced, based on the information gathered. Pinpointing the actual bug is the final step, providing there is correct information gained from the previous steps.

Woodpecker, a system that ferrets bugs out of trees, appears as a program-editor, displaying a menu of commands. The system is fully pattern directed. Graph-like trees are constructed to represent programs [FOUE84]. Each graph has the property that it contains one and only one entry point, and only one exit point. Such a graph, as shown in Figure 2.4, is called a spindle. Several variations of spindles can be created, representing a number of decision trees within the program. The tree is built up of these spindles.

One advantage to this representation is that much reasoning about programs involves searching for paths that exist or don't exist. This searching normally can be a time consuming process, but it already has been taken care of with this representation. The trees are independent of the programming language, because they are semantic rather than syntactic trees.

There are a few operations that Woodpecker relies on. Woodpecker has the ability to rename a variable that originally was used for more than one purpose, for example, local variables used in different procedures for unique jobs. It may also permute statements to add understanding to the role of a variable. This statement exchange also can be applied to any two primary spindles, or subtrees, that come off the main spindle. Auxiliary variables that

```
1  N := N0 + 1
2  N := N - 1
3  AGAIN := 0
4  FORALL J FROM 1 TO N
5  IF A(J + 1) LESS THAN A(J) THEN
6  B := A(J)
7  A(J) := A(J + 1)
8  A(J + 1) := B
9  AGAIN := 1
10 ENDIF
11 ENDLOOP
12 IF AGAIN NOTEQUAL 0 GOTO 2
```

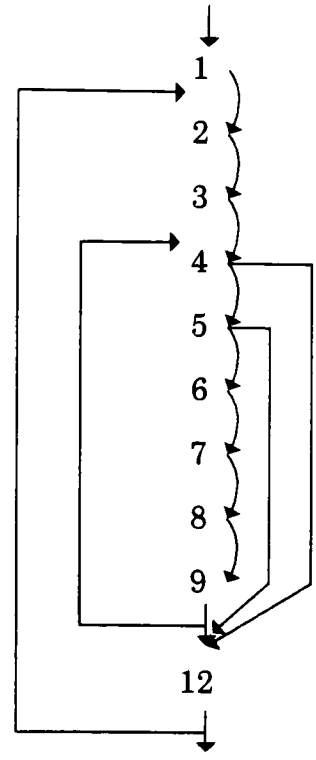


Fig. 2.4 Translating a simple program into a graph or spindle.

have been introduced are replaced by their definition to improve the meaning of the program. Loops can be suppressed and replaced with a recurrent equation.

These transformation tools work well on scalar variables; arrays, on the other hand, are a more delicate matter. Another problem is recursive programming languages, which lead to an internal representation that is not a finite tree.

The expert system for Woodpecker is implemented with metaknowledge that expands through the use of examples. A kernel is built around incrementally stored knowledge. The system originally was developed for use with a language called Gosseyn [FOUE84].

Knowledge is conveyed to the Gosseyn system through rules, written in almost natural language. The left hand side of a rule is a conjunction of propositions, introduced by "IF" or "FOR ALL". The right hand side is also a conjunction of propositions, with occasional calls to programmed procedures. The rules may express facts or openly express control. While rewriting the program as trees, an hierarchical representation is created in a working zone using a series of variables and pointers. the system then uses this hierarchical representation for programs, enabling it to "understand" complex parts of the program it is studying.

Knowledge-Based Programming Support Tool

While it is difficult to span the entire development of a software project, the Knowledge-Based Programming Support Tool seems to cover all possible categories of the process. The tool was designed to support all aspects of coding with the "capability of aiding programmers in various phases of program production such as design, coding, debugging, and testing" [HARA85] . Each unit that was developed also can be completely stand-alone. These various parts include: the knowledge-based programming assistant (KBPA), the design aid unit, the coding aid unit, the testing aid unit (TAU), debugging aid unit, and the knowledge acquisition unit (KAU). The interaction between these tools is shown in Figure 2.5.

The supervisory monitor of the KBPA handles the overall system. The design aid unit is a knowledge-based interactive system design aid that can be used by system analysts and non-expert system designers to construct software systems. It has knowledge of schematic system design and various domains of application. The design model is based on the data flow diagram method and conducts a constructive dialogue with the user. A coding aid unit provides a program editor and a design coder. Common data structures, algorithms, and an intelligent error recovery system are provided by the program editor, while the design coder builds program templates corresponding to the designed procedural layouts.

The debugging aid unit operates both on a shallow or deep level model and has a program analyzer. On a shallow level, the unit works on a set of

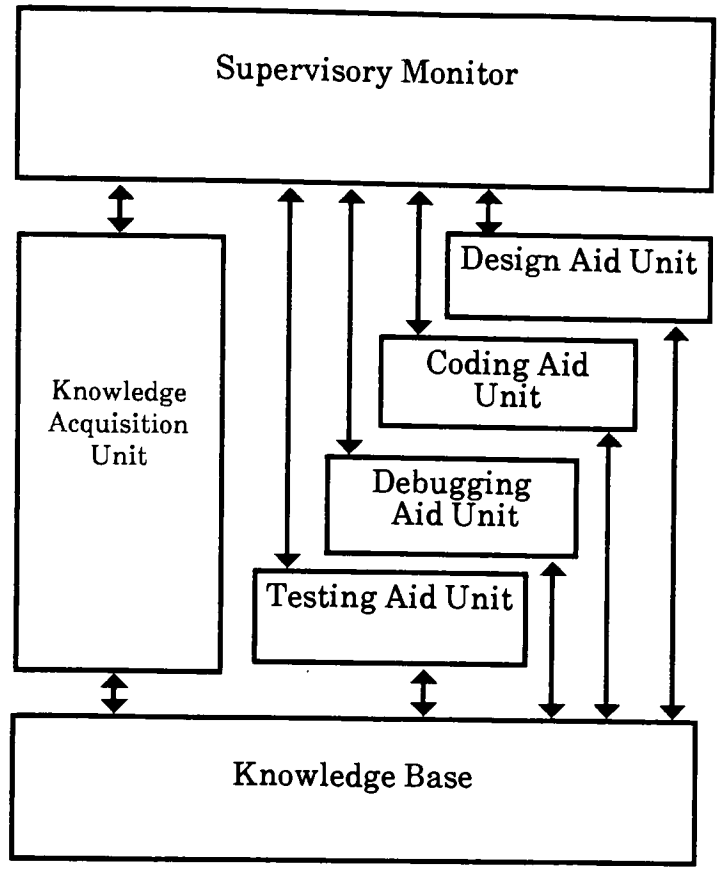


Figure 2.5. The overall structure of the KBPA.

situation action rules for compile and run-time errors. Situations represent the symptoms and the facts, and the actions are a description of the probable cause [HARA85]. Logic errors are the deeper level of debugging. Such logic errors are when the program terminates prematurely, exceeds run-time resources, and proffers incorrect results. The program analyzer is a supportive component that provides information on the data and control flow analysis. The knowledge acquisition unit is several tools acting together to facilitate the acquisition and modification of knowledge.

As a whole, the tool can operate in several modes: as a consultant / advisor, where the user seeks aid, as a knowledge base management system that answers queries, as a tutor that provides explanations and descriptive text, and as a learning device that learns by examples and mistakes.

Expert System / Application Generator

The ES/AG (Expert System / Application Generator) is a software engineering environment for the construction of intelligent application systems [CRON85]. It also covers the range of the entire development of a software project, especially generation tools and run-time problem solving tools. The generation tools consist largely of a first level AL (A-Level) compiler, which reads the application terminology component and determines attributes to be associated with the symbols and rules, and a second level (SL) compiler, which reads an application specific knowledge base, creates code and rule actions, and extracts information on the problem solving model structure. Semantic validation of the attributes and value associations to the symbols are provided. The generation tools also allow queries to come into the knowledge base in a formatted display.

Run-time problem-solving tools are broken up into several categories. Non-procedural control strategies keep track of the state of the problem-solving process, forward and backward chaining, and backtracking in two phases. Phase one is the invocation of procedure blocks, and phase two is the selection and invocation of rules. Explanation tools help provide the construction of dynamic dependencies during backward chaining and can be used to provide the reasons to a question of 'why'. A user interface to the problem-solving model through the user dialog module provides a set of capabilities for the end-user and it allows questions, input value validation, default values, help ability, string construction, and output routines.

A retrieval of application-attribute knowledge enables run-time access of attribute knowledge from a knowledge base and user-direct control of knowledge retrieval. There is an interface to the user-supplied program components, which can be connected to easily integrate user-supplied components to existing software. An interface to two Lisp interpreters, XLisp and Franz Lisp, allows calls to be included within the specifications. A list processing module is provided with a set of routines to access, change and create values of lists or list elements to avoid the overhead of a full LISP interface. There is an interactive debugger to access command level functions during system execution. It examines and modifies the rule selection ordering, lists all the symbols as calculated, and lists the rules as they are fired.

Approximate string matching is also provided with automatic recognition of abbreviations or misspelled information entered by the end-user, with a choice of possible replacements for the string.

2.1.3 Related Xerox Projects

While none of the Xerox tools or articles reviewed below incorporate artificial intelligence, many of them are used in the current software development process. The software development process has undergone upgrades and reviews, which were discussed in section 1.2. Some of the tools used in this process will be reviewed in depth below, specifically the System Modeler, DFTool, and Release Tool. A base reference for the software development steps in preparing a link relies heavily on the Describe File Tool software (DFTool) [HOWTO85].

The System Modeler

Xerox's Computer Science Laboratory at Palo Alto Research Center (PARC) has developed a System Modeler for a distributed environment [LAMP83], which provides automatic support for several different kinds of program development cycles in the Cedar programming environment. It uses information stored in a system model that describes the versions of various files, the interconnections between files, any additional information needed to compile and load the system, and hints on the locations of the files needed.

The Modeler is capable of a variety of operations on the system. It implements the representation of the system, tracks changes made by the programmer, automatically builds an executable version of the system, and provides complete support for the integration of packages as part of a release.

To facilitate the interaction with the various machines and file services, the Modeler maintains several files that facilitate reduced search and compile times. The top model is a list of models and the file server and directory where each can be located. An object type table and projection table are accelerators for the Modeler, providing a list of object types and entries requiring recompilation for the next iteration of the system. Two types of delays exist in such a distributed environment. First, if the file is on a remote machine, it must be found. Second, once found, it must be retrieved.

The Modeler provides a well integrated system for a distributed environment. It handles version control, remote interaction with other machines and file servers, retrieval of needed files from these remote devices, storage of the resulting compiled system, and creation of a release of the specified system model. Although it does not handle the diagnosis of possible errors in each of its operations, the Modeler utilizes the distributed environment to its full advantage.

Describe File Software Tool -- DFTool

The Describe File software has the ability to describe the version and location of files, relieving the user of needing to know where the files are located and which versions of the files to use. A single description file, or DF File, may describe all required files in a program. Thus, the name of that DF File could be used simply to bring over all files needed to work on a specific program. After those files have been modified, the DF program will store back only those files that were changed. Importing and exporting of explicit files allows for careful sharing of programs between implementors. Describing a software system with a DF file enables the use of the tools for managing files, verifying program consistency, and allowing programs to be a part of a major software release.

Four of the most used DF (describe file) programs are BringOver, SModel, VerifyDF, and DFTool. BringOver retrieves the files listed in a DF file from their remote file servers, possibly overwriting different versions already on the local disk. It insures that all files for a component, and the correct versions of those files, are on the local disk. SModel stores changed versions of files back on remote file servers and produces a new DF file containing references to the newest versions. Normally, the new DF file is also stored remotely for use by clients of that component. VerifyDF checks that a DF file is complete and consistent, that is, that all files needed to build the top-level object files of a component are listed in the DF file and are consistent with regard to the Mesa compiler and binder. DFTool provides a window interface to the other DF programs.

The DF file that accomplishes these functions usually has three sections: the interface or implementation files needed by the client, the list of files that comprise the component, and the imported files needed to build the component. The files exported by the DF software are marked by the keyword *Exports*; other files that are part of the DF file are marked with *Directory*; and the imported files have the keyword *Imports*. An example of a DF file is included in Appendix A. A standard use of the DF programs for a specific module follows a basic sequence: BringOver the DF file to insure that all files needed are on the local disk in the correct version; modify and test the files; SModel the DF file to store back the changed files and update the DF file to reflect the new versions; VerifyDF of the DF file to verify that the files are complete and consistent. This last function is not often used, causing problems to go unnoticed until the link. These basic programs along with a specified set of options provide a sound basis for version checking and change control for a multi-user software project.

A DFTool Example

One Xerox software group took it upon itself as a project to develop a system that efficiently created a basic file management system [FILE86]. The project report provides a clear definition of the contents of a software package released for test. The system was divided into three parts: the creation and storage of a link, the storage of a recent release and everyday links, and the backup of old release links.

The first part of the report, which covers the creation and storage of a link, utilizes the Describe File tool (DF Tool) and helps to explain better the necessary information for correctly monitoring module versions [FILE86]. Instruction is also provided for floppy archival storage of software releases, including the naming of revisions. A version tracking system was installed to allow for a check-in and check-out of all files within the release. Such a librarian system prevents simultaneous changes to a single file, thereby eliminating revision editing problems.

Another Extensive DFTool Application

Within Xerox's Office Systems Division, the Mesa group implemented the DF system that provided file management with explicit version control. The

implementation proved to be very effective for managing versions of software in their distributed programming environment. It was used to assist in the development of the last version of the Xerox Development Environment, which included 501,442 lines of Mesa code in 5360 files [LEWIS].

Version control within this environment was a problem since each programmer worked on a separate Xerox Workstation linked by an Ethernet connection. Locating the correct versions of files in this network was a difficult problem for programmers. Developers would explicitly copy files between their workstations to modify, compile, build, and test their software. Mesa also used both interface files and implementation files to construct its components. The use of the DF system within this process provided a way of checking the correct versions of each necessary component.

In the past, the Mesa group managed the software with the use of three software tools: FTP, MakeA, and Include Checker. FTP (File Transfer Program) transfers files over the Ethernet between the local disk and a remote file server. FTP has an option to retrieve or store only newer versions of files. MakeA generates command files that do the retrieve-build-store steps for rebuilding a component. MakeA executes a script file written for the component. These scripts are programs in a simple language that includes variables, conditional statements, and primitives for interacting with the user. A MakeA script can be parameterized to allow the user, for example, to build all or only part of the component. The location from which files are to be retrieved also can be a parameter, allowing the user to build private versions of the component.

IncludeChecker checks a set of Mesa source and object files for consistency. It can be used to check the consistency of a single software component or an entire release. It also can check both local and remote files. The IncludeChecker also will generate a compiler and binder command that will do any rebuilding necessary to make the collection of files consistent.

Despite care, this system still allowed mistakes, the cost of which was high. Some of these mistakes included wrong versions of files being retrieved,

files not being stored, difficulty in retrieving other than the most recent versions, and extensive time required for IncludeChecker.

Difficulties were encountered with DF software in the early stages of implementation. These difficulties stemmed from the lack of practices that needed to be developed for the group to customize the DF process to itself. These practices included learning to run VerifyDF to catch version problems earlier, using a program librarian to handle multiple implementors for a component, and specifying a default remote directory to avoid wrong version usage on a local disk.

Definite advantages were discovered in using the DF software. Retrieval and storage of files is simpler and less error prone. VerifyDF provides a strong guarantee that a software component is consistent and that its DF includes all needed files. It is easier to maintain different versions of packages since each DF file is a snapshot of a package's files. DF files also provide a readable description of the software.

Release Tool

After a period of developing software, a group often will want to produce a release. A release is a complete set of consistent software saved in a known, safe location (i.e. the archive directory). The ReleaseTool checks the files on a given integration directory for consistency and completeness. It then copies the files to the archive directory and generates new description files that describe the release. Included in the description file is a CameFrom clause, which indicates that the files described have been released [CONDE85].

Once this tool has processed all the necessary files and copied them into the archive directory, these files then can be copied to the release directory. ReleaseTool does not handle this final transfer. The recommended movement of files between directories is:

Working -> Integration -> Archive -> Release.

Many times a release can be built upon a previous release, using the modified set of release files. In many cases a release coordinator is responsible for

overseeing the release process. The release coordinator's duties include: identifying the components to be included in the release; storing their files and description files on the integration directory using DFTool in SModel pre-release mode; verifying the local consistency and completeness of each component on the integration directory using VerifyDF; preparing a top-level description file that describes the components, as well as the release parameter files for ReleaseTool; running the ReleaseTool; correcting any problems; and at completion, copying the appropriate files to the public release directory.

The ReleaseTool has three phases. Phase one verifies the consistency of all description files to protect against version conflicts and to ensure that all files exist remotely. Phase two verifies the release's completeness through VerifyDF on all description files. Phase three transfers all files to the new locations and produces the new description files that describe them.

It is possible that phase one may need to be run several times to correct errors. Phase one generally needs to be executed again before proceeding on to phase three. Phase two can be run at the same time as phase one, although on a different machine due to space constraints. ReleaseTool operates with either a window or a command line interface. It requires a great deal of free space on a workstation to run, due to phases one and three's use of a BTree as a cache to record information about known files [CONDE85]. Various parameters also provide added instruction on what particular options of ReleaseTool will be used.

2.2 Languages and Tools

In order to better understand the implementation of AMUSED, it is necessary to take a closer look at the language, tools, and system on which AMUSED is based. While the actual implementation language is inconsequential, it bears mentioning since much of the current implementation is the result of the features of this language and the tools.

2.2.1 Common Lisp

Common Lisp, a recent dialect of Lisp, is a successor to MacLisp. Its development was influenced by ZetaLisp, as well as by Scheme and InterLisp [STEELE84]. Common Lisp was developed to meet a set of eight goals. These goals are: commonality, portability, consistency, expressiveness, compatibility, efficiency, power, and stability.

Commonality

In order to privide a common dialect of Lisp, Common Lisp was designed to merge a set of variations of Lisp in a similar direction for the basic functions. From this base set, extensions could be added to accomodate any particular inconsistencies associated with various environments.

Portability

Common Lisp intentionally excludes those features that cannot be implemented easily on a broad selection of machines. Features that are difficult or expensive to implement are either avoided altogether or implemented in alternative ways of implementing these features are developed. Other features which normally might only be available only for select versions of Lisp, are either avoided or made optional. Common Lisp is designed not to rely on the machine specific requirements, but on the general definition of the language, while still offering a variety of implementation options.

Consistency

The definition of Common Lisp avoids inconsistencies between the interpreter and the compiler by maintaining that both the interpreter and the compiler produce identical semantics on correct programs as much as possible.

Expressiveness

Only those expressions that have been proven to be the most useful and understandable have been retained the language definition. Awkward or confusing expressions have been excluded.

Compatibility

ZetaLisp, MacLisp, and InterLisp is the primary order with which Common Lisp strives to be compatible in its implementation. These versions of Lisp are among the most common in the Lisp community. Common Lisp plans to maintain compatibility with these versions without sacrificing most general forms of compatibility with other versions of Lisp.

Efficiency

High-quality compiled code is a top priority. Common Lisp has added a number of features to facilitate more efficient compiled code. A high degree of efficiency can be achieved through the use of properly compiled code. S-1 Lisp, one implementation of Lisp, already has a compiler that produces code for numerical computations that is competitive in its execution speed with code produced by a Fortran compiler [STEELE84].

Power

MacLisp traditionally placed emphasis on providing system-building tools. These traits were passed on to Common Lisp. It is this emphasis that may in turn be used to build the user-level packages that are provided by InterLisp. These packages would be placed on top of the current definition of Common Lisp and do not currently exist.

Stability

It is intended that CommonLisp will change only slowly and with just cause. The various dialects that are supersets of Common Lisp may serve as laboratories within which to test language extensions, but such extensions will be added to Common Lisp only after careful examination and experimentation [STEELE84].

2.2.2 TMYCIN / EMYCIN

The tool that is responsible for the actual functionality of AMUSED determines, for the most part, the actual operational capacity of the system. A number of tools were reviewed for redundant use before TMYCIN was selected.

2.2.2.1 EMYCIN

One of the first shells to be put to general use, EMYCIN, essential MYCIN, was derived from the MYCIN medical expert system by removing the application specific knowledge [TANI87]. It is a rule-based system with a complete explanation facility built in, one of its greatest strengths. Through the use of automatic bookkeeping [TANI87], the editor monitors the changes made by the user and records pertinent information. If a rule is added or changed, the date and user's name are stored with the rule for later reference. Such information is useful in the case of multiple experts working on a single system.

Developed at Stanford University in Interlisp, EMYCIN contains a rigid backward chaining control mechanism which limits the application to diagnosis and classification type problems [WATE86]. Rules have the IF antecedent THEN consequent form. The antecedent is a collection of true or false expressions; the consequent contains the conclusion that follows from the antecedent. A context tree organizes the EMYCIN objects in a simple heirarchy, which provides some of the inheritance characteristics of a frame system. The inference engine works through a list of rules to find the answer, or it exhausts the list. If the list is exhausted, the system asks the user for a value.

EMYCIN attaches certainty values ranging from -1 (false) to +1 (true), to every expression within the antecedent [WATE86]. Each expression is determined to be true if the certainty value is above a given threshold (for example 0.2); it is false if the certainty value is below a given threshold (for example -0.2). Special evidence combining formulas are used to decide how the certainties of the antecedent are to be combined [WATE86]. The resultant certainty value is used to update the certainty value of the consequent.

Human engineering appears to be one of the greatest strengths of EMYCIN [HAYE83]. It offers a convenient environment and user interface. The knowledge acquisition facility permits the rapid construction of a knowledge base, and explanation facilities for the system are also well defined. A keyword parser is built in to aid the user, searching for key phrases and words such as HOW and WHY.

The constrained control structure of EMYCIN is one of its major weaknesses. Only those items listed in the context definition are valid subjects for the antecedent and consequent expressions. Expansion of the context is difficult, making the extensibility of the system limited. With such close context checking within the rules, the efficiency of the system is raised to a high level. Backward chaining (consequent-driven rules) is the principal method of control, but there is a limited provision for forward chaining (antecedent-driven rules). It is these restrictive inference methods that have sacrificed generality to gain efficiency.

The data format for a specific context is described in a context class. A context contains a name of the context record (usually the name of the object being identified or diagnosed, e.g., ROCK or PATIENT), the list of parameter descriptions, the initial data, which is a list of parameter names whose values are to be entered at the start of every consultation, and the goals, a list of parameters whose values are sought as the result of the consultation. An example context definition is shown below [NOVAK].

```
(defcontext  'rock                                  ; Context name
             '((color (brown black white))          ; Parameters
              (hardness posnumb)
              (environment (igneous metamorphic sedimentary)
                 "What is the type of geologic environment
                  in which the specimen was found?")
              (identity atom)
              (pretty nil))                         ; Nil for yes/no
             '(color)                               ; Initial data
             '(identity))                           ; Goals
```

The parameters field allows for two additional entries where further information may be supplied upon user request. The first field prompts for the type of answer being requested (e.g. (brown black white)). The second field is

an optional question that rephrases the original question if the user still requires more information. Both fields are only accessed from a specific user enquiry.

Rules may be defined to the system singly or in groups. Each rule has a rulename, a premise and a conclusion. Usually the premise, or antecedent, is a conjunction of conditions grouped within a call to the function $AND and the conclusion contains a call to the function CONCLUDE. The DO-ALL function also can be used for multiple actions in the consequent. An example of a rule definition is shown below [NOVAK].

```
(defrules
   (rule101 ($and (same cntxt color black)
                  (notsame cntxt pretty yes)
                  ($or   (between* (val1 cntxt hardness) 3 5)
                         (same cntxt environment sedimentary)))
            (conclude cntxt identity coal tally 400)))
```

2.2.2.2 TMYCIN

TMYCIN, "Tiny eMYCIN", is a simple expert system tool patterned after EMYCIN. It was originally written by Gordon S. Novak of the University of Texas at Austin [NOVAK]. Recently, the tool was rehosted by several people to a number of implementations of Common Lisp for the Xerox environment. TMYCIN offers the same basic rule-based system, and constrained control structure as EMYCIN. The rule and context format is identical to that of EMYCIN. The certainty values contained in the EMYCIN antecedent and consequent also are maintained.

Written in Common Lisp, the underlying code for TMYCIN contains no more than the basic similarities to EMYCIN. The module organization of TMYCIN is shown in Figures 2.6 and 2.7 on pages 35 and 36. Doconsult, the main function of CIN (see Figure 2.6), controls all operations on the rule base. It is supported by a number of smaller functions which are responsible for controlling the backtracking and rule checking. The actual conclusion processing is executed within the Conclude function, shown in Figure 2.7. The feature set supported by TMYCIN is drastically reduced from EMYCIN's user friendly system. The basic context is the same, but the user interface features found in EMYCIN have been left out of TMYCIN. Only the basic input for

knowledge acquisition is supported. Also omitted is the natural language parser to identify key words and phrases from English sentences. The storage of date and user information attached to new and altered rules in EMYCIN has been omitted. Aimed at smaller applications with only one expert, this feature is no longer of primary importance to TMYCIN. Only a basic combining formula for certainty values was retained. Certainty value formula selection based on context and system features is not available in TMYCIN.

User interface facilities are limited. User interaction is restricted to key words, such as yes, no, how, why, and whynot. The user entries are not case sensitive. Multiple answers to enquiries are possible, allowing the user to allocate a degree of certainty for each answer. However, the format for such a reply is based strictly on the format required by the implementing language. Such an input format provides a poor interface to the user and is only described in a short tool document designed for the implementor and not for the general user. No on line help facilities for input suggestions are available.

The error handling for the system is also poorly designed. No mistakes in rule or context syntax are allowed. Error messages are not descriptive and are not recoverable. Re-execution of the tool is required to eliminate the error.

2.2.3 Discarded Tools

A few available languages and shells were considered in determining the tool to be used for the expert system. The tools were evaluated for ease of development, accessibility during use, development time, and suitability for the desired application. Prolog on the Pyramid computer system in the Graduate Computer Science Center at Rochester Institute of Technology was eliminated for several reasons. Access to the system for development was possible, but use of the resulting system by the Xerox programmer would be made extremely difficult. Modem connections, user-account access and user-account limits would restrict access to the Pyramid.

Quintus Prolog, recently made available on the Xerox workstations, was also an impractical solution. The Prolog system runs on top of a Lisp volume and is currently unavailable within Digital Systems division. Along with the

slow execution times, because of the Lisp base, support for Quintus Prolog, if obtainable, would be limited. Also, no one in the immediate Xerox work group was familiar with the system for use or maintenance.

The main user interface development environment in the Digital Systems Division (DSD) at Xerox, Interlisp-D, was likewise eliminated because of the amount of effort required to produce even a small amount of functional code. The overhead in time and code would have greatly increased the effort needed to complete the project. The TMYCIN tool offered the best options of an expert system shell without overpowering the small system with unwanted features. TMYCIN was originally written in Common Lisp to run on the Koto version of Interlisp, but the environment was no longer being supported. In order to obtain use of the TMYCIN tool, it was necessary to port TMYCIN to the Medley release of Common Lisp on the Xerox workstation, where it required a few minor adjustments.

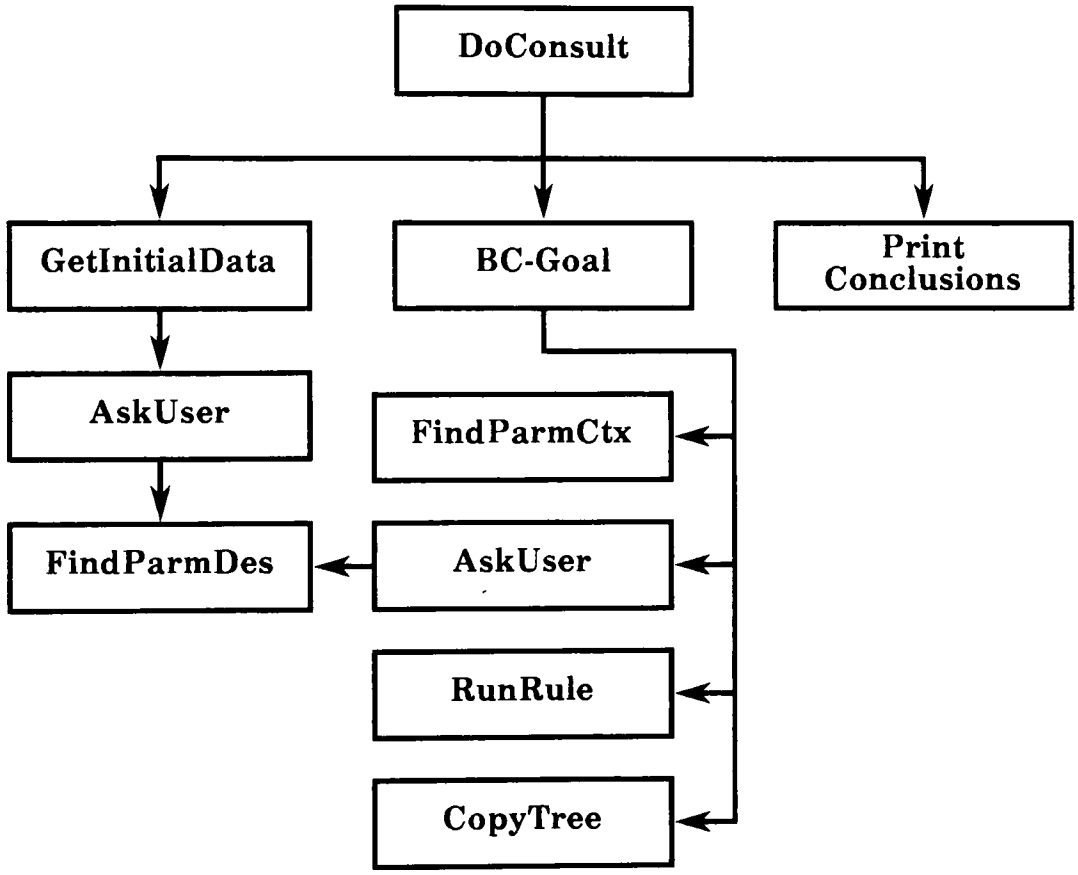


Figure 2.6 TMYCIN High Level Design

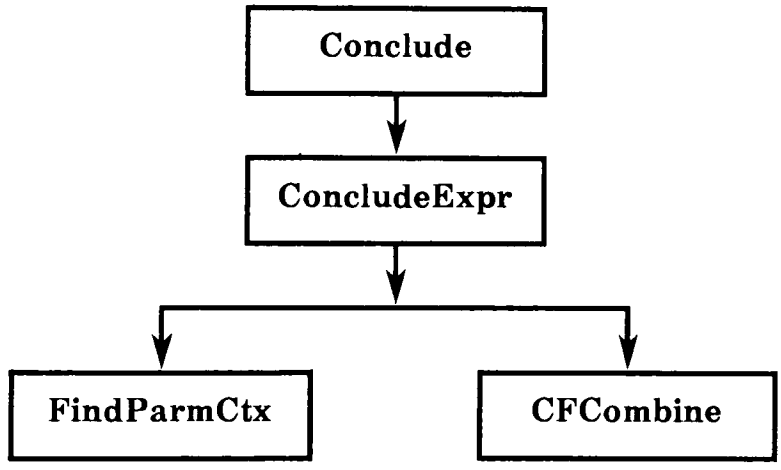


Figure 2.7 TMYCIN High Level Design (cont.)

2.3 Present System at Xerox

To gain a better understanding of where AMUSED fits into the software development cycle, a short description of the current hardware network at Xerox is provided. From this description, the AMUSED system is explained along with its current restrictions.

Most programmers in Xerox's software environment operates a Xerox 8010 or Xerox 6085 processor, which is a personal computer workstation with bitmap display, a two or three-button mouse, and a 42-megabyte or 80-megabyte rigid disk. These computers are linked by an Ethernet network to other workstations and also to shared file, print, and communication services. The local internet currently includes more than 350 personal workstations and 20 file servers. Within the local corporation in Henrietta and Webster, a collection of over 1000 personal workstations and 65 file servers are in operation on interconnected networks. This internet is only part of the worldwide Ethernet system, but this project only focuses on this local section of the network.

Locating correct versions of files within this computer network can be a difficult problem for programmers. File servers provide permanent file storage and respond to requests for storage or retrieval of files. Code developers explicitly copy files between their workstations and file servers while building and testing their software for ease of transfer and backup security. Each file server typically stores a dozen or more file drawers, containing several thousand files in several hundred different directories. There can be any number of versions of a given file in a directory, since older versions are not automatically deleted when a file is stored. The different versions, although bearing the same name, are unique due to their creation dates and a file version number assigned by the server.

No central directory exists to map file versions to their remote locations. Also, it is usually unreasonable to search for a file by enumerating each directory on each file server, since the file servers can only enumerate a few files a second. New developers frequently report that discovering which file

servers and directories contain the files they need is one of the most difficult parts of learning their job.

The Mesa language, other Xerox in-house programming languages, and related support tools increase the importance of version control. Mesa supports two kinds of modules: *interface* modules (or *interfaces*), which define abstractions, and *implementation* modules, which implement and use interfaces. To ensure that strong type checking is extended across module boundaries, Mesa requires that all references to modules be consistent (agree exactly in version). A number of languages, including ADA, use the same basic approach to version control, providing warnings on version mismatches.

Version checking has proven invaluable in organizing and developing large software systems. Strict version checking, however, has made another problem more significant: locating and using the correct versions of files. This problem is greatly complicated by the large number of intermodule references in a multi-user project environment.

Within the Xerox environment during the software development process, the use of an expert system has not been attempted previously. Many problems with the software development process are currently handled by a software tools support representative from the software tools group. When a problem arises, this representative is contacted, the problem is described or shown, and corrections are attempted. Through the collection of data from the software representatives, estimates on the success rate of the representative method as described below are quite high. For each reported problem, a solution of some degree is found.

Trial and error may often describe the support person's task when addressing a new problem. If the first correction does not work, more information is gathered. Further attempts are made until the problem is defined fully and resolved satisfactorily. Each problem is resolved to a positive solution or defined as unresolvable at the current time. If a problem has no immediate positive solution, then a workaround is found to accomplish the same objective.

With the current system, several problems exist. Only one representative is available to debug problems for all users in the entire copier / duplicator software community at Xerox. This community usually consists of at least a dozen separate software projects. Use of a dedicated representative for problem solving can become expensive in terms of dollars and man-hours. Because of this difficulty, turnaround time on the resolution of a problem could take anywhere from several hours to several days, depending on problem complexity, problem backlog, and the priority of the problems in the queue, as well as the priority of the project with the problems.

Due to the extended turnaround time for solving a problem, the development cost and time for a given project increases. Added cost to the project is also incurred to help support a dedicated support person. This dedicated expert gathers knowledge about the problems that arise during that person's time as representative. After a time, the support duties change hands. Attention to this stage can be critical in preventing the loss of acquired knowledge. Incorrectly transmitted information can lead to improperly diagnosed problems, or lost information. During the transition of representatives, time also can be lost due to the lack of experience of the new representative. Over time, this domain experience grows, based on the knowledge acquired by previous representatives and actual experience.

One of the additional difficulties with problems that are not caught early on is time delays. If a link is incorrect and the software code needs to be prepared again, the release of the linked files could be up to a day late, depending on the size of the project. Some problems can be fixed by modifying the resulting link files before testing begins. These modifications, or patches, require the solution to be placed into the source at a later date.

3. Implementation

AMUSED was developed to assist software support to projects by providing an easily accessible tool that can be used by individual programmers. Although the system may not have all the answers to problems encountered, it will eliminate the need for dedicated support for simpler problems, as well as provide additional information to the representative that eliminates some possible causes of the problem.

The AMUSED system runs on top of the Medley version of Common Lisp on the Xerox workstation. Underlying the AMUSED system is the TMYCIN tool, which provides the shell base required of AMUSED. Several additions to the TMYCIN base were added which will be discussed in more detail in the following sections. Among the additions are the acceptance of global information from a specified source. The addition of global data acceptance expands TMYCIN's constrained context structure to a more usable level. New terms and questions can be added without risking the context control of the inner shell. It is this addition which made TMYCIN usable for this project. Without the global expansion, the TMYCIN tool is too restrictive to properly handle the problems encountered.

AMUSED's operation may be looked at from several levels. To explore properly all of the functionality, it is necessary to view the implementation from three basic levels: the appearance, the functionality from a user's perspective, and a technical description of the operation. In this way, all levels and features of the system will be examined.

3.1 System Overview

The AMUSED diagnostic tool is constructed of an extensible layered shell expert system. TMYCIN, the inner core of AMUSED, can be called with a varied selection of rule categories. TMYCIN provides the rule-based shell, a set of terms in a local context for error checking, and the user prompted information. It also provides a facility for calculating a value

based on the certainty of the user's answers. AMUSED blends both the local context information of TMYCIN and the global domain-specific questions and variables of the system. Facilities are provided for checking global variables and accessing a set of global question databases, which will be explained in more detail in later sections.

3.2 AMUSED Appearances

When in the idle condition, the AMUSED tool is represented by an icon as shown in Figure 3.1. Opening this icon, or tiny window, reveals the main AMUSED window. The window consists of three basic portions, shown in Figure 3.2. At the top is a two line title bar which shows the AMUSED acronym and the acronym expansion upon the second line. The second section is the command line, offering access to three commands -- *Diagnose*, *Assist*, and *Done*. Largest of the three regions is the interaction window. It is in this window that the interaction with the user takes place. To accommodate any amount of data that might be transmitted through this window, the text scrolls upward one line at a time as required.

Figure 3.1 AMUSED Icon

The three commands that are presented in the main AMUSED window provide access to three different operations. *Diagnose* is the access to the main diagnosis function. It provides an interactive question-answer session to resolve the errors found during the linking process. Implementation of the diagnose tool and its actual operation will be discussed in the next section. *Assist* provides a set of simple descriptions of specific topics referred to in the Diagnose tool. These descriptions answer the commonly asked questions about various topics and provide information on where to turn to for further information. Further information on the operation and flexibility of *Assist* is provided in following sections. *Done* is

the command which exits the AMUSED operation by closing related files and resetting the tool.

Figure 3.2 The AMUSED window

3.3 User Tutorial

AMUSED is an interactive tool designed for those people responsible for creating the executable software releases for a programming project. Specifically, AMUSED provides assistance for the link process in the software development process, as well as the transfer of source files to and from remote sites to the link workstation. Below is a tutorial description of how to operate AMUSED and what the user sees during each step of execution.

3.3.1 Diagnose Walkthrough

To enter the diagnosis mode of AMUSED, the user selects *DIAGNOSE* from the command line. An option window appears above the AMUSED window, accompanied by a prompt message in the main interaction area. Before the actual diagnosis can take place, the options in the attached window must be set to reflect the current options to be diagnosed.

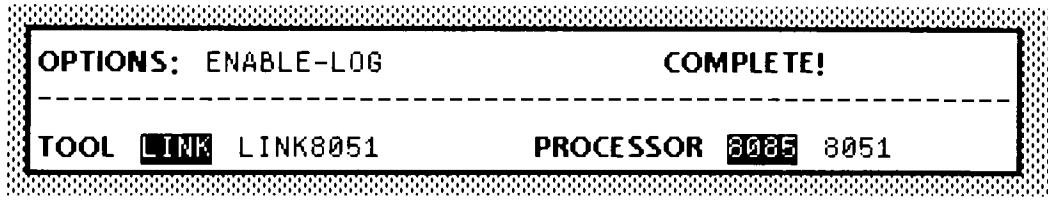


Figure 3.3 Diagnose Options

The options to be set before diagnosis, shown in Figure 3.3, include the link tool and the microprocessor responsible for the link problems. An Enable-log option is available which enable the recording of the diagnosing session for future reference. This feature will be discussed in more detail in a later section. Once the options have been correctly set, *COMPLETE!* is selected. This command signals the tool that the user is ready to continue.

Several general questions are asked at the start of every diagnosis session. These questions help to specify the category of the error. A sample of one of these general questions is shown in Figure 3.4.

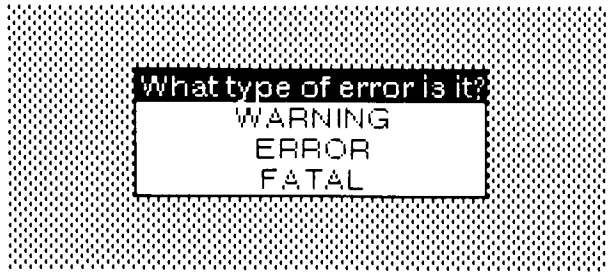


Figure 3.4 Selecting the problem category

Following the general questions, a rule base specific to the type of error is loaded. A statement is printed to the interaction window, informing the user of this action. The user is then asked a series of questions which AMUSED uses to determine the source of the problem. A dual format is used for the questions. The question is printed to the interaction window as well as appearing in the top bar of a popup menu, the contents of which are the list of possible answers. A sample of this dual format is shown in Figure 3.5.

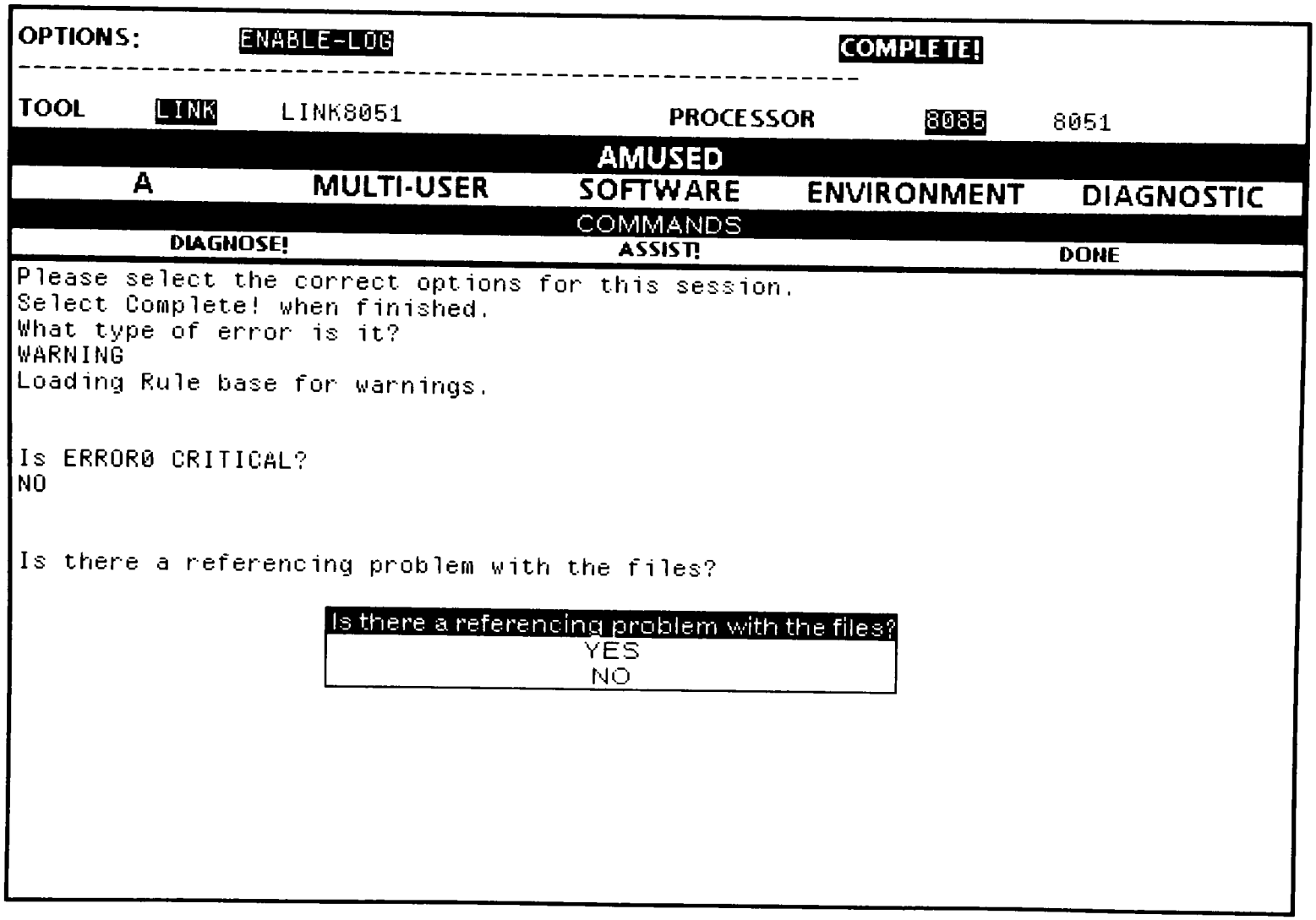


Figure 3.5 A Diagnosis in progress.

If sufficient information has been gathered to determine the problem, then a description of the problem and advice for correction of the problem are printed out. The format for the conclusion is shown in Figure 3.6. A key phrase summarizing the problem and a value representing the certainty of the answer concludes the diagnosis session.

Figure 3.6 A completed Diagnosis, asking to continue with another diagnosis.

Should a solution not be found from the questions presented, the user is prompted for a problem statement which fits the set of questions as presented. The user-supplied answer then becomes the solution of the problem. This set of conditions is then stored for future use, allowing the system to gain knowledge about its domain.

When the Enable-Log option is set in the the options window of Diagnose mode, all statements, questions and answers that are printed to the main AMUSED window are saved for future reference. At the end of the diagnose session, a file containing these statements is written out. The name of the file is then reported to the user for their use.

Upon completion of the diagnosis, the user is asked whether another diagnosis is desired. If affirmation is given, the tool is reinitialized and the diagnosis process begins again. Otherwise, the option window is closed, the location of the log file is printed, and AMUSED returns to the idle state, waiting for a user selection.

3.3.2 Assist Walkthrough

The Assist function offers users on-line help and descriptions for the most common problems encountered in the diagnose process. To obtain assistance from AMUSED, ASSIST is selected from the main command line. A menu appears offering a selection of general categories for which assistance is available. The base menu is shown in Figure 3.7.

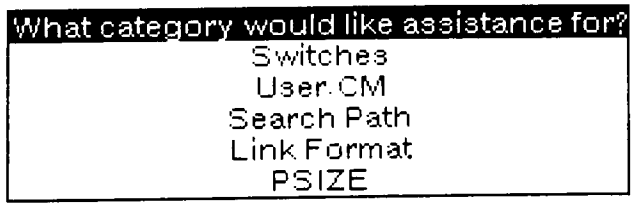


Figure 3.7 Initial Assist Menu

After one of the general categories is chosen, another menu appears offering a selection of specific topics under the chosen category. An example of the second menu category is shown in Figure 3.8.

Figure 3.8 Secondary Assist menu after selecting the switches topic.

Following a selection in the sub-category menu, a simple description is printed to the interaction window.

The descriptions that are printed out for each specific topic include a brief description of the topic, shown in Figure 3.9. These descriptions may be a short term definition or an outline of how a particular command is to be used. It may also list the possible options available and their meanings. A reference to the tool manual where more information is obtainable is also listed.

Figure 3.9 Description supplied after Invalid Switch Combinations is selected

3.4 Technical Overview

AMUSED is implemented using a rule-based expert system. Since the number of possible errors is a finite number and the error messages can cover more than one probable solution, the problems lend themselves easily to goal driven reasoning based on sets of rules and facts. The best answer is selected from the choices available, and the system does not make any unwarranted guesses. A simple explanation facility is available to help the user understand how to prevent certain problems from recurring. The instruction facility includes such items as the proper use of the tools, the correct setup of files needed for these tools, advice on version control, or additional tools available for use.

An interactive question and answer format is used to lead the programmer to provide enough information to determine the actual cause of the problem, assist the user in fixing the problem, and instruct the programmer on how to prevent the problem's recurrence. This expert system provides the capability for a problem to be solved quickly and with lower debugging cost than in the current environment. Added features also include instruction on the usage of the tools of the system and explanation of the problems encountered. AMUSED is loaded into the Medley version of the Common Lisp environment and executed from its own window.

3.4.1 Basic Control Flow

AMUSED's flow of control begins in the top AMUSED shell. A pictorial description of the control and data flows can be found in Diagram 3.1 on page 61. The user interacts with the top shell and the TMYCIN shell through the main interaction window. From this window the user may select from two main operations, *Diagnose* or *Assist*. The third operation, *Done*, halts the AMUSED program and closes off any opened files.

The diagnose operation begins by opening an option window which needs to be correctly set for the diagnosis before continuing. To signal the correctness of the options, *COMPLETE!* is selected from the options window.

The options are then reviewed for compatibility and registered as global variables within the system. These options determine which particular question database will be loaded and also narrow the choice of the initial rule base to be used. An additional question is asked to determine the initial rule base before the actual diagnosis begins. Asking this error type question before the diagnosis reduces the rule set to a manageable size and increases the response time substantially.

Upon the loading of the rule base, control is then transferred to the interior TMYCIN shell. It is from the TMYCIN shell that the basic inferencing and interfacing is done. TMYCIN asks the user a series of questions through popup menus and the user interface window. These questions collect the local and global information which will match the rules in search of the correct solution.

Local information within the rules is checked against the TMYCIN context of acceptable facts. Global information, such as certain rule-specific questions and options set in the diagnose options window, is not subject to the restriction of the local context. The absence of such a restriction allows new information to be stored by the user for future reference. It also allows more flexibility of the rules and information as the tools are updated and improved.

3.4.2 Diagnose Options

As described in the user tutorial, upon selection of the Diagnose command, an options window appears above the main AMUSED window. Options that can be set in the option window include: Enable-log, the tool used and the processor. These options, previously shown in Figure 3.3 on page 44, must be correctly set for the diagnosis before continuing. By correctly set, it is meant that there is a limited degree of error checking of the options selected in the window. Both the Link and Link8051 tool are only compatible with a certain microprocessors or controllers. Not all of the processors are implemented in this version of the tool. Selecting an incompatible tool and processor results in a warning and an opportunity to reselect the options. However, there is no way for AMUSED to know if a

legal but incorrect selection is made. AMUSED cannot account for human error.

Once the options are selected, selection of *COMPLETE!* signals the correction of the options. These options are then translated into a series of global variables that will be used in the selection of the correct rules as well as the final solution. The correct question database is loaded according to the linking tool specified. The linking tool along with the error type, which is requested immediately following the option selection, determine which rule base should be loaded for the current diagnosis session.

The Enable-Log option allows a separate file to be opened. From that point on, all material that is printed to the main AMUSED window is also printed to the log file. Upon completion of the diagnosis, the name and location of the file is printed out to allow the user access to a documented account of the diagnosis.

3.4.3 Initial Questions

Mention was made in the previous section of a question being asked of the user before the actual diagnosis takes place. Currently, only one question is asked before the diagnosis. The question concerns the actual error type being diagnosed. The categories of Warning, Error, and Fatal which are shown in a popup menu similar to Figure 3.10 are actually the categories of errors generated by the Link and Link8051 tools. It is this categorization of the problem being solved which allows AMUSED to select a subset of the rule base to be accessed.

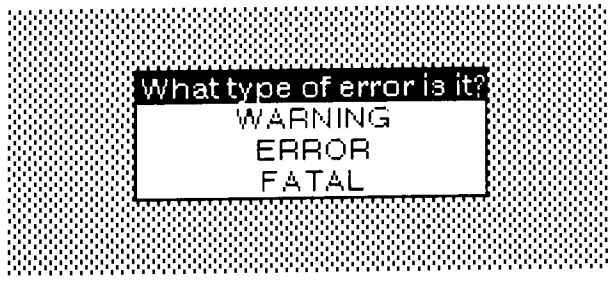


Figure 3.10 Selecting the problem category

Selection of a subset of the actual rule base has proven to be a crucial element when dealing with the response time of the tool. The actual time variations and further information of the rule base categorization will be discussed at a later date.

3.4.4 Asking Questions

Once the global variables have been set and the selected rule base and question database have been loaded, the diagnosis is ready to begin. TMYCIN now takes over control of the tool. It is responsible for chaining through the rules, presenting the questions, and processing the answers. This updated version of TMYCIN was expanded to handle two types of questions, both local and global questions.

The local questions are those questions which must have a topic contained in the TMYCIN context of the expert system. These questions can be either yes/no, numeric, or short answer, depending upon the question desired and the response specified in the context. User assistance about the question is also available from the context. Each category specified in the context can specify two levels of prompting for the user, providing a better understanding of what is being asked. These local questions can be identified by listing the name of the actual instance being diagnosed, as shown in Figure 3.11. In this example, ERROR306 is the actual numbered instance of the problem being diagnosed.

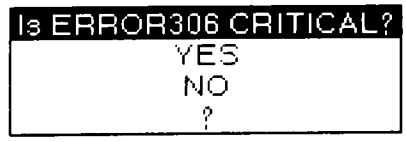


Figure 3.11 A local question

All other questions are considered global, since they are not checked by the local context. Two varieties of global questions are used, those that involve a global variable and those that ask a specific question. Global variable questions check the state of the global variable against a desired value. These questions are conditions existing within each rule. The user is

never asked about the state of these variables since they were set in the options window. It is these states which guarantee that the rule is specific to a certain tool. Similar problems with different tools generate unique rules.

Specific questions are listed in the rule conditions as a reference into the question database. These references are translated into actual yes/no questions which are presented to the user. The results from these questions are matched against the expected answer within the actual rule being tested. An example of a specific global question can be seen in Figure 3.12.

Figure 3.12 Specific global question

All interactive questions are presented to the user in the form of a popup menu appearing at the current cursor position. The menu shows the question in the title, and the possible answers below it. Only a selection within the popup menu will satisfy the question. Any mouse selections outside the popup menu will result in the popup menu being redisplayed, waiting for a correct response. Questions and the user-selected answers are also displayed in the main AMUSED window for a number of reasons. Users may find it more readable to read the question found in the AMUSED window. The question is also printed to the AMUSED window to store the decision path for those users wishing to preserve the path for review during the diagnosis as well as for those people desiring a log file.

Some questions still require the user to input their reply using the keyboard. In these cases, the AMUSED window is able to accept input from the keyboard as necessary. Only short answers are accepted by the system. Input of a carriage return ends the input string.

3.4.5 Processing Questions

TMYCIN uses a forward chaining rule walk-through approach. For those rules currently available to the system, a top-down approach is taken to search for a rule that has acceptable conditions. Once a rule is found in which the first condition is accepted, then the conditions on that rule are checked further. If the conditions match identically, then the conclusion is executed. However, should one of the conditions not pass, then TMYCIN will back out of that rule and continue checking conditions further down the list of available rules.

Should none of the rules pass the conditions, then TMYCIN will prompt the user for a solution to the current problem. The local and global conditions that have been tested to that point are then saved as a new rule for the system. In this way the TMYCIN system learns to expand its knowledge base. The learning techniques will discussed in a later section.

When a given rule fails one of its conditions, the next rule in the list of active rules is checked. If there are duplicate conditions in the following rule, these conditions are valued as indicated earlier by the user. Asking duplicate questions should be avoided whenever possible. Local conditions are saved within TMYCIN, but the same is not true for all global conditions.

Global variables are stored within the system, requiring no need to prompt the user for these values. The questions which are extracted from the question database must be marked in some manner to prevent duplication also. To prevent unwarranted duplication of these global questions, every question which is presented to the use is kept on a list of questions previously asked.

The global question itself is not stored, but the key information used to retrieve the question, as well as the answer and certainty factor for the answer are stored on the list of questions asked. The correct procedure is then to check for the current question, in the list of questions that have been asked. If the question has been asked, then the previous values are used in the current rule condition. If the question is not found on the list of questions previously asked, then the question is presented to the user.

Several structures have been implemented in the AMUSED system which are used by the TMYCIN shell, and several structures are unique to AMUSED. These structures include certainty factors, the distribution of the rules into assorted rule bases, and the implementation of a question database for specific questions.

3.4.6 Certainty Factors

Reference was made in several places to certainty factors being used in TMYCIN. Certainty factors are used to calculate the degree of confidence of a solution which is derived from the user's input. There are two variations for collecting these certainty factors from the user. The most simple form is in the standard yes/no question. If the answer is yes, then the certainty factor is 1.0; if no, then the certainty factor is -1.0. For specific global questions being asked, another popup menu appears after an answer is selected. The menu asks the user of their confidence of the answer they just gave. The menu offers 5 different selections to which are attached set certainty factors for the listed English phrases (see Figure 3.13). These certainty factors are assigned as follows: Not certain (0.0), Somewhat Certain (0.5), Fairly Certain (0.7), Very Sure (0.9), Positive (1.0).

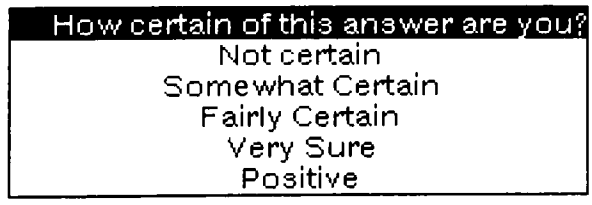


Figure 3.13

The calculation of the resulting certainty factor is a simple averaging of the values collected from the conditions within the rule.

3.4.7 Distributed Rule Bases

Currently, the system has six separate rule bases available. Three rule bases are available for Link problems, and the remaining three are available for Link8051 problems. Each of these rule bases for each tool are

for problems which are classified to be of type Warning, Error, or Fatal. The decision to separate the rules into six separate rule bases was done for several reasons.

The projected number of rules for the AMUSED system is about 300. For this number of rules, it was determined that the execution time of the system would be unacceptably slow. The breakup of the rules into these six categories increases execution time significantly. Maintenance of the smaller rule bases is much simpler and easier to understand.

It is also possible that even the distribution of the rules into six portions might not mean an acceptable speed up in execution time. Although this does not currently appear to be the case, a contingency plan is in place to further split the largest rule base into subcategories based on the particular topic of the error. This proposed split would increase the number of rule bases from 6 to 16 with two of the rule bases being virtually empty. Those two empty rule bases would be the error rule bases which would further split into five sub-rule bases. The distribution of the current and proposed rule bases is show in Diagrams 3.2 and 3.3 on pages 62 and 63.

The actual implementation of the individual rule bases is simple. The rules contained within the given rule base are specific to the particular tool and error type with little chance of a rule for another tool or error type being stored in the wrong rule base. Should the system be broken down into the sub-categories for the Error rule base, then the chance of duplication increases slightly for those problems which do not fit easily into a specific category. In these cases, it is possible that the rule or rules for that problem would be duplicated amongst the necessary sub-categories in order to duplicate every possible line of reasoning in detecting a problem.

Since there is little or no duplication of problems between the rule bases, the chance that a user might actually traverse down the wrong line of reasoning is relatively small. It is possible that the user might answer the initial questions incorrectly or that the options might be set improperly for the problem to be dealt with, but these problems cannot be easily compensated for by the AMUSED program. AMUSED cannot deal with

what the user is thinking, only with that information which is provided for it. In the expanded rule base system, the inability to cross between rule bases is compensated for with the duplication of questionable rules and problems in multiple categories. The duplication of a limited number of rules prevents serious slow downs of the AMUSED system, such as those processing slow downs which would occur when AMUSED switched to another rule base to do its checking.

3.4.8 Question Database

Within the AMUSED system, there is the need to address questions that are specific to the problem at hand. The user needs to verify whether certain files or names are correct. Each problem needs to check its own set of parameters which is not possible with the original TMYCIN shell. Thus the global strategy for AMUSED has been added. By adding access to system variables and specific global questions, the extensibility of the AMUSED system has become much larger. From the current strategy, it is possible to add or delete conditions and parameters from the rules with relative ease. Such a versatile strategy is not possible with the TMYCIN shell alone.

With the great diversity of possible conditions and questions which can be implemented in the AMUSED system, some organization is necessary to ensure that the access times to this information do not sacrifice the speed with which the user requires the information. This means that concerns for system consistency must also be addressed. The speed and the extensibility of the system are both preserved with the organization of the question database into two databases based on the tool use. These two databases are then broken up into sub-categories ordered by general topic.

Currently, two question databases are provided to the AMUSED system. Each database contains questions specific to its own tool, one for Link and the other for Link8051. Within these two databases, the information is distributed into information based on the general topics that are covered. Each of these topics is marked with a key word for easy information retrieval. A list of questions is contained in each topic, marked by a key number for quick retrieval. The key words for the categories currently in the system are as follows:

FILE - File problems
CONTENT - File content and syntax problems
OPTION - Problems with the options specified
SPACE - Space allocation problems generated by the files used by the tool or by the tool itself.
MISC Other miscellaneous problems
WARN Warnings
FATAL Fatal errors

The general organization of the database can be represented by the following:

(KEYWORD ({(Question.Number "Question")}*))

for example,

```
(FILE       ((1 "Question 1")
             (2 "Question 2")
         ...
             (N "Question N"))).
```

Each of the databases is stored as a global list of sub-lists. For one of the question databases to be active, it is reassigned to the global variable GLOBAL.QUESTION.DATABASE. It is this global variable which simplifies the code to one implementation rather than two. Searching for specific questions is done through this global variable. An associative search on the question's keyword is done to localize the search to a specific category. From this localized section of the database, another assocative search on the question number is done to retrieve the question intended.

Since the information retrieval is done using an associative search on a value, the order of the categories or the questions is not important. Therefore, questions can be added or deleted without disturbing the other questions within the list. Question numbers are not order dependent. Instead, the specific question is assigned a number as a key, not an index.

3.4.9 Assist Mode

As described in Section 3.3.2, the Assist function operates through a collection of menu selections. The selection at the first level of menus directly selects the second menu to be displayed. A second menu selection determines the actual text to be displayed. A unique function of print statements exists for every topic available. These individual functions allow for easy updating of descriptions as well as a versatility in the presentation arrangement of the topics.

3.5 Summary

AMUSED is a functional system which allow users to find solutions to problems during the linking process. Information is provided to supplement the solutions given. References to the correct Link user manual also are provided to lend further assistance to the problem. While only a prototype system, AMUSED has the capability of adapting to new errors and situations, thereby maintaining its usefulness to the user in a changing environment of software development. A number of possible enhancements and adjustments could be made to the system, which will be discussed in Chapter 5.

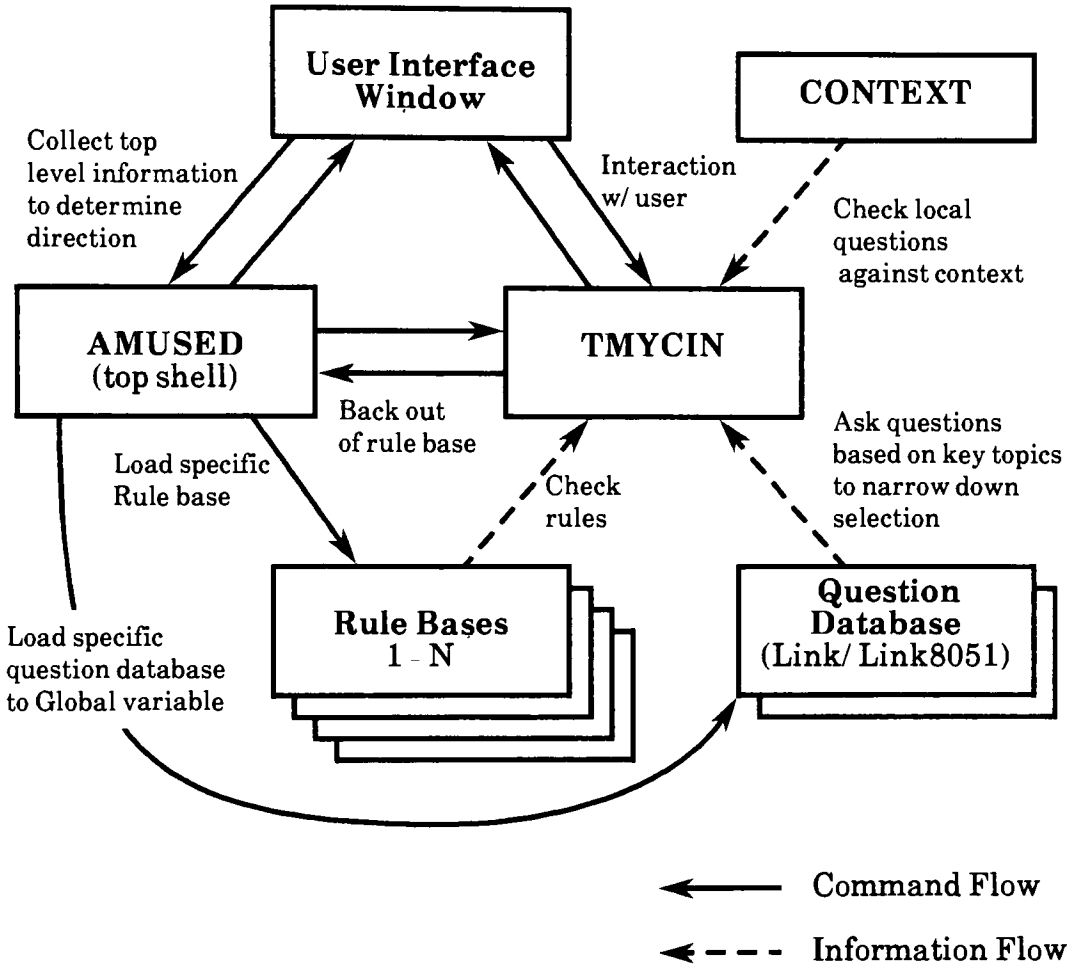
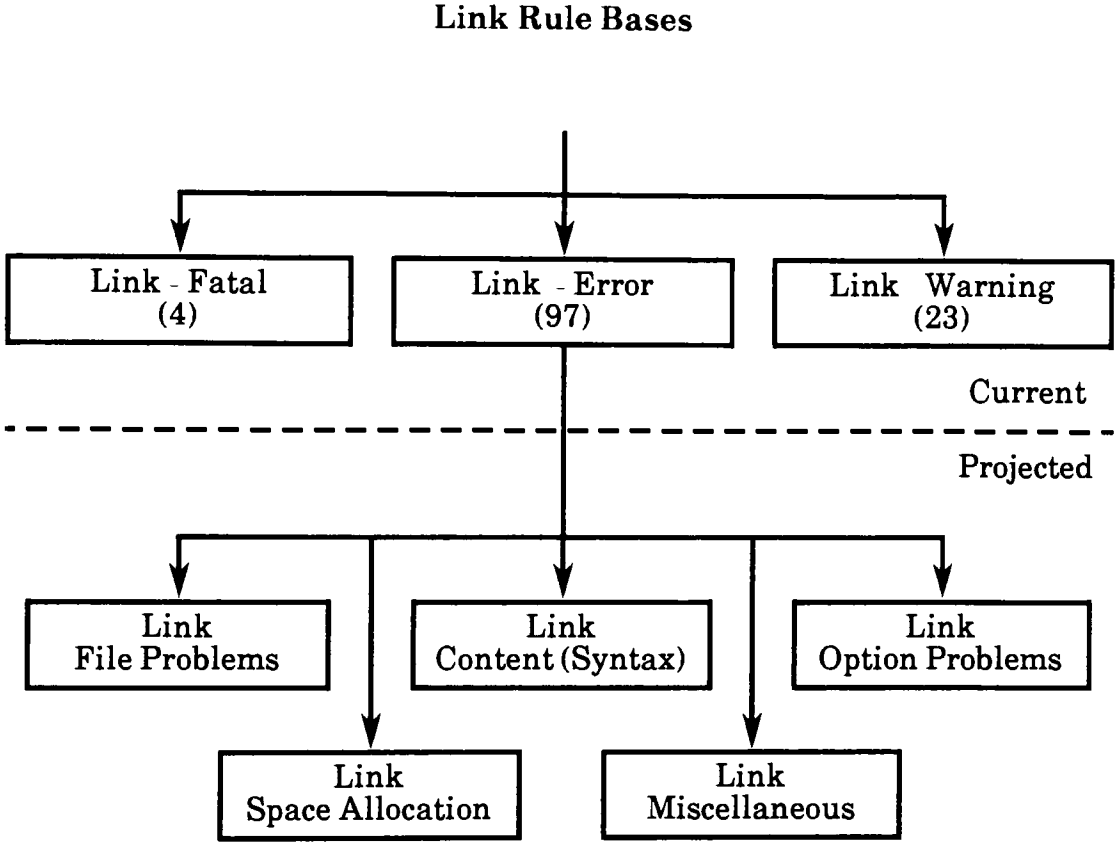
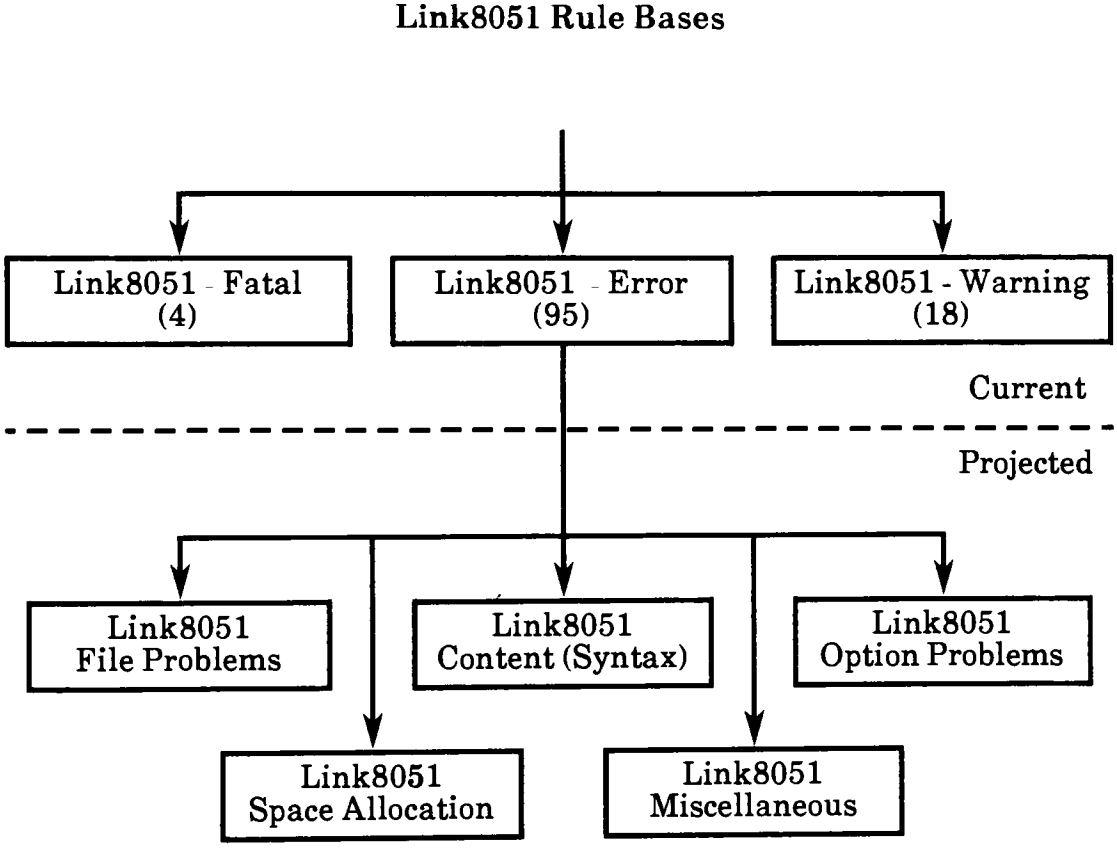


Diagram 3.1 AMUSED Flow of Control



(xx) - Estimated number of rules per Rule Base

Diagram 3.2 Rule Base Distribution



(xx) - Estimated number of rules per Rule Base

Diagram 3.3 Rule Base Distribution

4 Testing
4.1 Data Collection

During the collection of errors to be input into the system, it was possible to contact a number of representatives from various products within Xerox. The range of projects represented included the low-volume, mid-volume and high mid-volume business products. In these projects several different software languages were used for development and spanned from assembly code to several high-level in-house programming languages. Although the developing language varied from project to project, all projects which were contacted used Link or Link8051 to produce their executable code for testing and development.

From these varied projects, a wide array of problems were encountered and entered into the system. Several patterns were recognized in relation to a number of problems that frequently arise in the development process. Better testing conditions were created for these errors than for errors that occur infrequently.

Not all of the possible errors were encountered during the relatively short sampling period. This lack of a broader selection of errors may be due to the actual state of maturity of each of the projects and the tools themselves. Many errors found in the tools were encountered during the early stages of tool development or in the early stages of any given project. More errors are encountered during the initial stages of any project until the programmers become familiar with the syntax and protocol of the environment.

Although the error set for AMUSED is not complete, those errors that are most commonly encountered are stored in the system. In this way, even a limited system can be used for common problems. A number of errors were also added to the system by researching entries in the consultant's digest, an electronic log of user problems with the tools and the programmer's code. A number of errors with sufficient information were found in this reference as well.

4.2 Early Implementation Results

When the TMYCIN tool was first brought up, many problems were encountered with making the CML Common Lisp hack work on the Koto release of Interlisp. It was discovered that the hack was no longer supported and several revisions had occurred, making TMYCIN extremely difficult to debug. Since the tool was written in Common Lisp, the decision was made to port the TMYCIN tool over to the Xerox release of Common Lisp, commonly referred to by its revision name, Medley.

The new Medley Common Lisp environment was almost entirely different from the Interlisp environment of Koto. It became necessary to relearn the storage and loading of files as well as some of the basic organization of the fundamental data structures underlying Lisp. Once again, support and documentation were in short supply as only a few people at Xerox in Henrietta and Webster had ever used the Medley environment.

With the Xerox Common Lisp environment came the implementation of packages. All functions are associated with a given package of software, whether it is Xerox Common Lisp (XCL), Common Lisp (CL), or Interlisp (IL). Whenever a function is used that is not in the currently set default package, the package name must be attached to the function or variable name. These package manipulations caused problems in the manipulation of data retrieved and sent to the user, as well as the readability of the code.

Due to the package considerations, the comments placed in the functions disappeared from any hardcopies printed until the correct options were placed in the files. This problem was not one listed or solved in the documentation, but one that had to be discovered from another Medley user.

There were difficulties in the expansion of the TMYCIN tool to incorporate global variable setting within the rule base, as well as additional functions which needed to be added to incorporate this global status. The TMYCIN tool was also expanded to accept questions which were specific to the line of reasoning being followed by the rules. To incorporate the use of specific questions, a question database related to the error messages was added. This question database is accessible through the rules being executed.

4.3 Testing

The AMUSED system that underwent testing consisted of 23 rules for both Link and Link8051 distributed into six rule bases, and two question databases containing 55 questions. Although the number of rules appears to be only a fraction of the estimated 300 rules projected for the completed system, operation of all rules is the same. Addition of the remaining rules will not change the function of AMUSED, only the rule base and question database size and execution time.

Three stages of operation were tested on AMUSED. The first prototype was brought up with a single rule base consisting of 25 rules. Execution times were noticeably slow. Print statements could be seen writing out each character across the line.

Distribution of rules into six unique rule bases was then implemented. Regulation of what rules were loaded and available with each diagnosis was also added. Execution of this revised implementation was much faster, although still somewhat sluggish at times.

As a final stage, all AMUSED functions were properly compiled, increasing the performance significantly. Even after increasing the number of rules, no significant decrease in execution speed was noted. If a decrease in system performance appears once the full system is implemented, the further distribution of rules into the 16 rule bases can be implemented. This distribution was described in Section 3.

Noticeable improvements in understandability and performance were found with the implementation of the popup menus for questions. Users were less likely to make incorrect decisions, although they were somewhat intimidated by the menus' inability to disappear until a correct selection was made. One drawback noted by users is the inability to stop the diagnosis once begun, to get assistance on a specific topic. This option would be possible were it not for the global implementation of specific questions. It does not allow the user to interrupt the execution once it has begun.

5. Conclusion

From the studies of results of AMUSED testing, as well as projected results of a fully implemented system, it can be seen that the AMUSED system has a definite effect on the issues of cost, efficiency and compatibility. With the implementation of the AMUSED system within Xerox, the number of problem calls, especially those related to minor problems and easily diagnosed problems, will decrease. This predicted decrease in support calls could reduce the cost of a dedicated support person or allow them to respond to more complicated problems more quickly. More time might even be allotted to different tasks.

A new task would need to be assigned to the support person, that of monitoring the new rules being generated. These rules would have to be validated and merged into the system. Updated versions of the rule and question databases within AMUSED could then be released, providing a more adept tool. In this way, the expert system could be kept current with latest system knowledge.

Along with a possible decrease in support costs, the problem solving turnaround time also would be reduced. Problems could be solved more quickly with AMUSED. Elimination of possible causes would be possible through AMUSED. Also, since the information of AMUSED is preserved, this resource could help reduce the initiation time of new support personnel. Reliable replies could be supplied to programmers by AMUSED while the new support person learned the tool and the debug process. Such a benefit reduces the transition of support people as well as cutting down the response time to the users.

5.1 AMUSED Shortcomings

AMUSED runs in Lisp, while most program developers work in the Xerox Development Environment (XDE) or on personal computers. No program development is done in Lisp. In order for a project to have the AMUSED system available, a dedicated Lisp volume or possible a dedicated Lisp workstation would be required. The Lisp volume could be on a user's workstation, but volume switching would be required whenever AMUSED was needed. A dedicated workstation would not have this problem.

Along with the inconvenience of a dedicated Lisp workstation could the problem of the cost of a dedicated workstation might exist for some programs. In this case, the AMUSED system would have to be accessed through the software support representative. If the system were running in the support area, then it could be accessed by a personal visit or a phone call to the support representative. Such a setup would decrease the effectiveness of AMUSED, but it would still prove beneficial to the software support team.

A recent drawback to the implementation of AMUSED is a change in direction within Xerox, causing the phase out of the use of Xerox workstations. Such a decision will mean a shift of AMUSED to a new environment for continued effectiveness. It could, however, mean that AMUSED could become an archival expert system, preserving past implementations of the tools while new implementations and new tools are being adopted. It is unclear what the future of AMUSED will be.

5.2 Alternate Approaches

Several different approaches could have been taken with AMUSED to attempt to reconcile the discussed shortcomings of the tool. The major approach difference for the tool would be writing the expert system in Mesa or another compatible language. Developing AMUSED in Mesa would allow its use within the XDE environment.

One major drawback to this approach included the reinvention of an expert system shell to handle AMUSED's need. This approach or a similar approach could be the most advantageous for Xerox; however, that approach would mean more work than should be required in developing an expert system and more than is needed for a master's thesis. The current AMUSED system possibly could be thought of as a prototype for a similar tool to be developed at a later date within Xerox.

It is also possible that a different expert system tool may exist that would be better suited to the local and global aspects that are now employed by AMUSED. In such a case, the tool may be one that is executable within XDE for easier programmer access.

5.3 Possible Extensions

With the current AMUSED tool, a number of extensions could be made to enhance both the error processing capability and the user interface functions. Some of these extensions include remote connection ability, an error log parser, and a natural language front end parser to better handle user input.

Remote connection ability would allow AMUSED to connect to another workstation or file service and either remote browse or retrieve the necessary files. These files could be specified by the user or implied from directory information supplied by the user. Once these files have been retrieved, the operator then could copy in actual sections of the error messages for processing. Some of the file that would be beneficial to acquire would include the error log files from links or the actual link listing from a link run. These two files would list the errors and warnings that occurred in the link.

Once these files are available on the AMUSED workstation, another extension could be employed. An error log parser could be added to parse through the specified error log and retrieve the errors within the file. The parser then would be able to take the errors found and parse the strings for key words to be supplied to AMUSED for diagnosis. The location of the errors would not be a difficult problem to solve but it would be would be a time consuming problem. Correct location of the errors in a file would require much testing to assure correct data is retrieved. A benefit to this approach would be fewer typographical mistakes to deal with on AMUSED's part, leading to less frequent occurrences of the tool being misled by mispellings.

The natural language parser that would be needed to parse the actual error messages would need to be extensive, looking for every possible message structure known. The system would be undergoing constant renovation for changes to the tools or new messages. Although the concept is a convenient one, the amount of time and work behind the approach might not make it a feasible enhancement.

Another natural language parser could be employed at the front end of AMUSED. This parser could be used to interpret user input and determine

what is being asked or what information is being provided to AMUSED. Key words or phrases could be triggers for finding the information supplied by the user. Making this approach feasible would also take a great deal of code and work to make even the simplest statements easy to use. A limited version of this approach might be possible by limiting the user's input to a subset of replies or key words.

In looking at the AMUSED system as a whole, the foundation has been laid for a tool that is useful to application programmers. Once the system has been supplied with a full set of rules and questions, the system will be beneficial to programmers on a wide variety of projects at Xerox. Use of several enhancements mentioned will aid the user's ability to interface with AMUSED and provide better output.

AMUSED References

[BOYD84]
Boyd, D.L., "Knowledge Based Programming: A Proposal for Research", *Third Annual International Phoenix Conference on Computers and Communications. 1984 Conference Proceedings*, 19-21 March, 1984, pp. 2-3.

[CONDE85]
Conde, Daniel, *Release Tools Reference Manual*, Xerox Corporation, Sept. 1985.

[CRON85]
Cronk, R.N., D.V. Zelinski, "ES/AG: System Generation Environment for Intelligent Application Software", *SOFTFAIR II. A Second Conference on Software Development Tools, Techniques, and Alternatives*, 2-5 Dec. 1985, pp. 96-100.

[DFSO85]
DF Software Reference Manual, Xerox Corp., April 1985.

[FILE86]
"File Management System", Xerox Leadership Through Quality Project report, August, 1986.

[FOUE84]
Fouet, J.M., "An Expert System for the Manipulation of Programs", *4th Jerusalem Conference on Information Technology (JCIT). Next Decade in Information Technology*, 21-25 May 1984, pp. 460-7.

[HARA83]
Harandi, M.T., "A Knowledge Based Programming Support Tool", *Proceedings of Trends and Applications 1983. Automating Intelligent Behavior. Applications and Frontiers*, 25-26 May, 1983, pp. 233-9.

[HARA85]
Harandi, M.T., "A Knowledge Based Design Aid for Software Systems", *SOFTFAIR II. A Second Conference on Software Development Tools, Techniques, and Alternatives*, 2-5 Dec. 1985, pp. 67-74.

[HAYE83]
Hayes-Roth, Fredericks, Donald A. Waterman, Douglas B. Lenat, *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.

[HOWTO85]
"How to Link", Xerox internal memo, December 13, 1985.

[KAISL86]
Kaisler, Stephen H., *Interlisp, the Language and Its Usage*, John Wiley and Sons, New York, 1986.

[KORE86]
Korel, Bogdan, "A Program Error Localization Expert System",*Proceedings of the SPIE International Society of Optical Engineers,* Vol. 635,1986, pp. 111-18.

[LAMP83]
Lampson, Butler W., Eric E. Schmidt, "Organizing Software in a Distributed Environment", Computer Science Laboratory, Xerox Palo Alto Research Center, 1983, pp. 1-13.

[LEWIS]
Lewis, Brian T., *Experience with a System for Controlling Software Versions In a Distributed Environment*, Xerox Corp., Palo Alto, CA.

[MANU84]
Manucci, Franco, "Expert Systems in Telecommunications Software Development Environments", *Proceedings COMPSAC 84, The IEEE Computer Society's 8th International Computer Software and Applications Conference*, 7-9 Nov. 1984, p. 386.

[MARC84]
Marca, D., "Software Manufacturing and Large Software Maintenance", *Digest of Papers Compcon Spring '84. Twenty-eighth IEEE Computer Society International Conference*, 27 Feb.-1March, 1984, pp. 312-315.

[NORD86]
Nordin, H., "Using Prototypes for Knowledge-Based Consultation and Teaching", *Proceedings of the SPIE International Society of Optical Engineers,* Vol. 635, 1986, pp. 402-7.

[NOVAK]
Novak, Jr., Gordon S., *TMYCIN Expert System Tool*, Computer Science Department, University of Texas at Austin, no date.

[SCHMIDT82]
Schmidt, Eric Emerson, "Controlling Large Software Development in a Distributed Environment", PhD Dissertation, UCBerkeley with Xerox, 1982.

[STEELE84]
Steele, Guy L., *Common Lisp, the Language*, Digital Press, Bedford, MA 1984.

[TANI87]
Tanimoto, Steven L., *The Elements of Artificial Intelligence*, Computer Science Press, Rockville, MD 1987.

[WATE86]
Waterman, Donald A., *A Guide to Expert Systems*, Addison-Wesley, Reading, MA 1986.

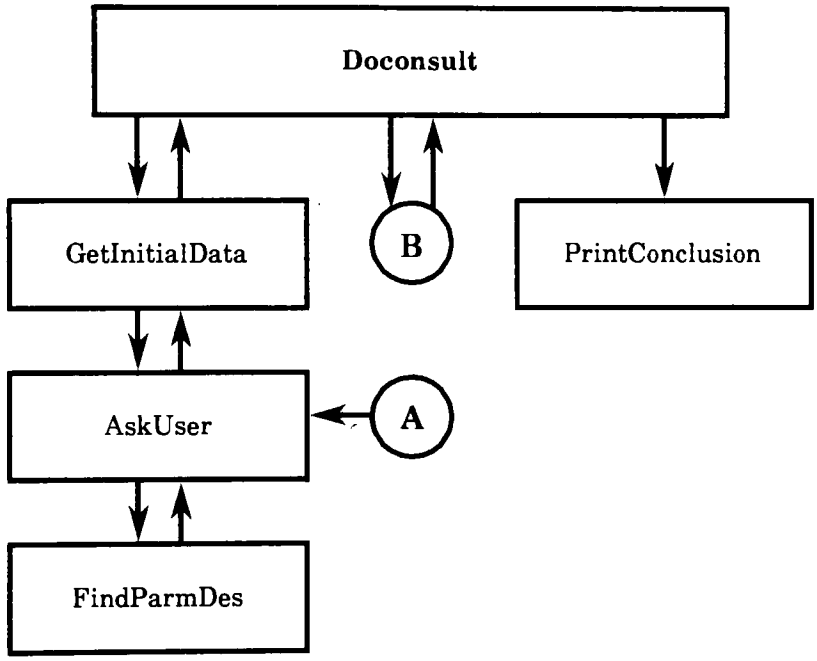
[WELI84]
Welin, C.W., "Expert Systems in Telecommunications Software Development Environments", *Proceedings COMPSAC 84. The IEEE Computer Society"s Eighth International Computer Software and Applications Conference*, 7-9 Nov. 1984, pp. 384-5.

Appendix A -- DF File Sample

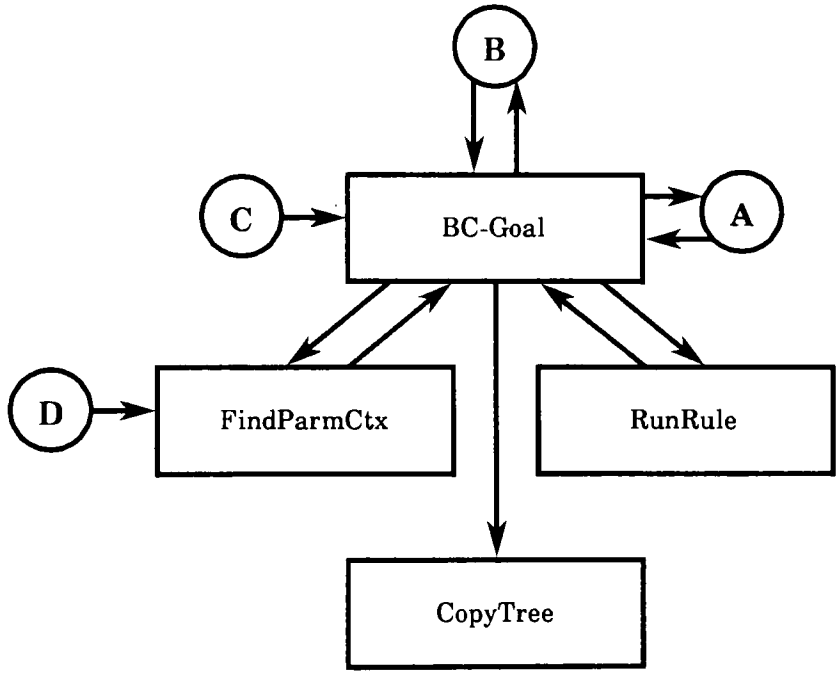
```
-- Compare.df        Last edited by Joe on 8-Feb-83 13:36:58

Exports [Igor]<Emerson>DF>     ReleaseAs [Idun]<APilot>DF>
    Compare.df                                22-Feb-83 13:59:40 PST

Exports [Igor]<Emerson>Compare>Public>  ReleaseAs
                                        [Idun]<APilot>Compare>
                                        Public>
    +Compare.bcd!18                     22-Feb-83 13:53:16 PST
    Compare.symbols!7                   22-Feb-83 13:53:18 PST

Directory [Igor]<Emerson>Compare>Private ReleaseAs
                                        [Idun]<APilot>Compare>
                                        Private>
    Compare.cm!2                        16-Nov-82 10:16:29 PST
    Compare.config!4                    16-Nov-82 10:21:06 PST
    CompareControl.bcd!13               22-Feb-83 13:50:09 PST
    CompareControl.mesa!9               22-Feb-83 13:49:53 PST
    CompareDefs.bcd!6                   16-Nov-83 11:48:21 PST
    CompareDefs.mesa!4                  16-Nov-83 10:16:47 PST
    CompareImpl.bcd!13                  18-Feb-83 14:55:08 PST
    CompareImpl.mesa!10                 18-Feb-83 14:55:00 PST
    CompareWindow.bcd!3                 23-Dec-83 13:52:41 PST
    CompareWindow.mesa!2                23-Dec-83 13:51:53 PST

Imports [Igor]<Emerson>DF>ComSoftPublic.df Of #
    Using [Ascii.bcd, Format.bcd, Heap.bcd, String.bcd, Time.bcd]

Imports [Igor]<Emerson>DF>MesaPublic.df Of #
    Using [Environment.bcd, Inline.bcd]

Imports [Igor]<Emerson>DF>FileSystemPublic.df Of #
    Using [MSegment.bcd, MStream.bcd]
```

```
Imports [Igor]<Emerson>DF>PilotPublic.df Of #
    Using [Process.bcd, Runtime.bcd, Stream.bcd, System.bcd,
    UserTerminal.bcd]

Imports [Igor]<Emerson>DF>TajoPublic.df Of #
    Using [Exec.bcd, FileName.bcd, FileTransfer.bcd, FormSW.bcd, Put.bcd,
           Tool.bcd, ToolWindow.bcd, UserInput.bcd, Version.bcd,
           Window.bcd]
```

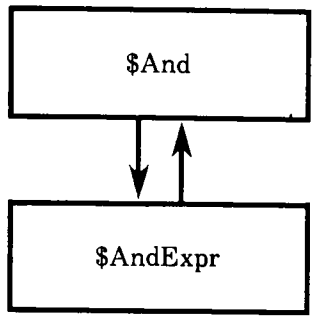
Appendix B -- TMYCIN Data Flow Diagrams

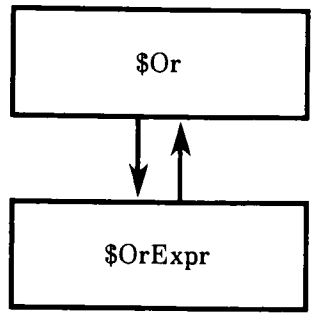


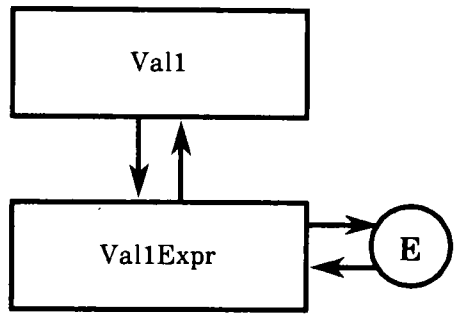
B.1

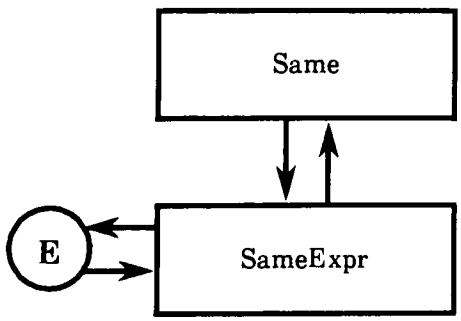


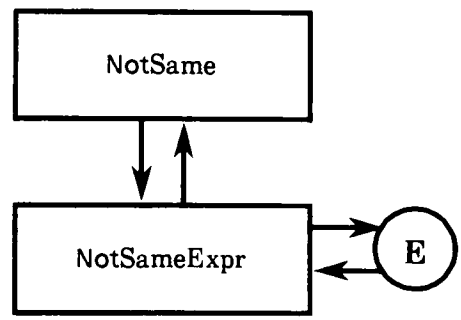
B.2


$And
$AndExpr

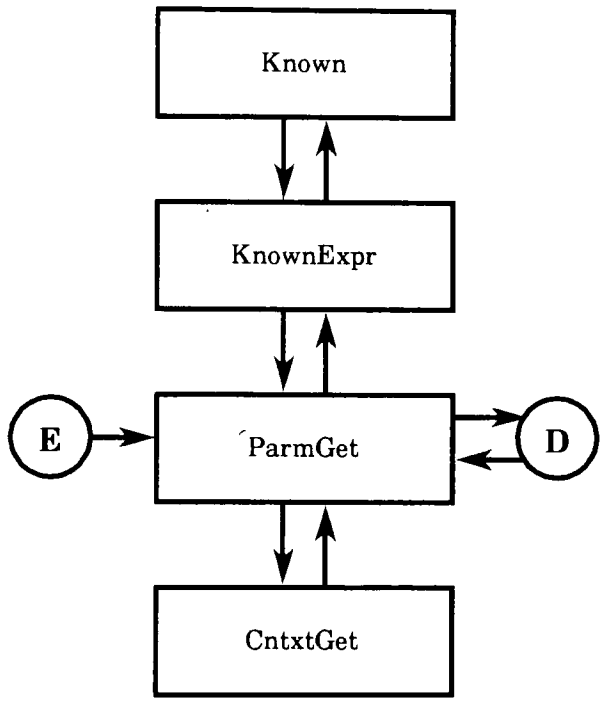

$Or
$OrExpr


Val1
Val1Expr
E

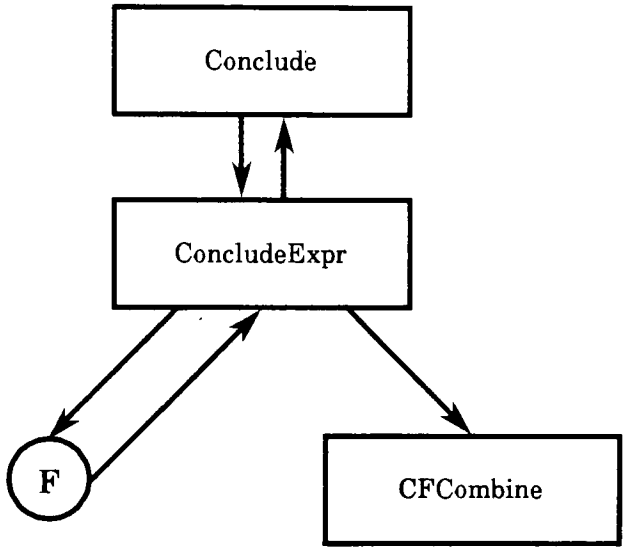

Same
E
SameExpr


NotSame
NotSameExpr
E

B.3



B.4

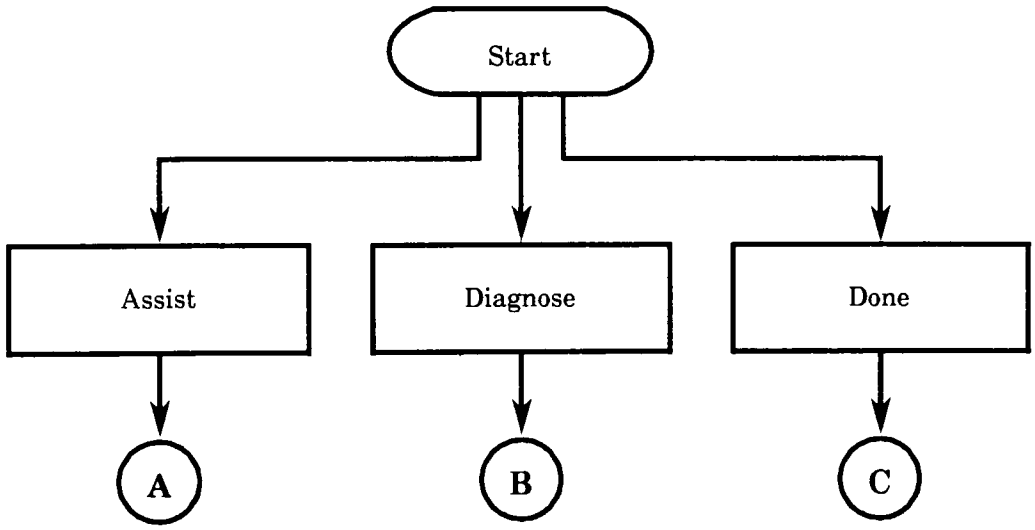


B.5

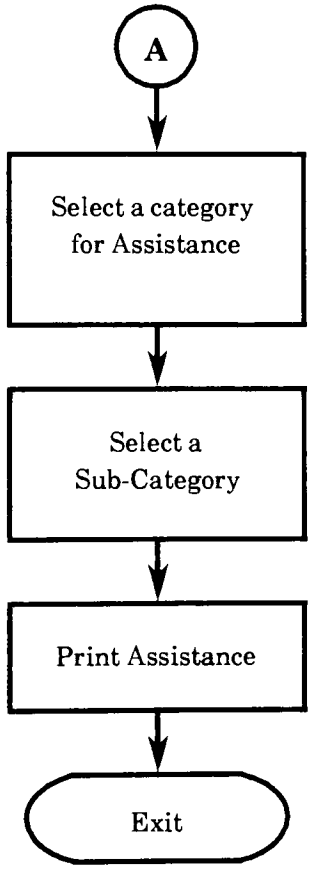
Appendix C --TMYCIN Function Description

DEFRULES	Calls DEFRULE and sets up each rule, checking for proper format. Each rule is added to the variable AllRules.
DEFCONTEXT	Sets up a data structure with the information given. The data structure contains the context name, a parameter list for all acceptable parameters, the initial data to look for (a starting point), and the goal parameters which need to be filled in.
DOCONSULT	The main function which runs the expert system, see Appendix B.
MAKE-CONTEXT	Sets a new data context class with a pointer to a parent context. Class ISA - Class PARENT - patient1
DO-ALL	Allows a list of conditions to be executed for the conclusion of a rule.
SHOWRULE	Prints out a rule in lisp format.
SHOWPROPS	Prints out the properties of a context.
ENGLRULE	Prints out a rule in English form. Calls PRINTPREMISES and PRINTCONC.
PRINTPREMISES	Prints out the conditions of a rule in English form.
PRINTCONC	Prints out the conclusion of a rule in English form.

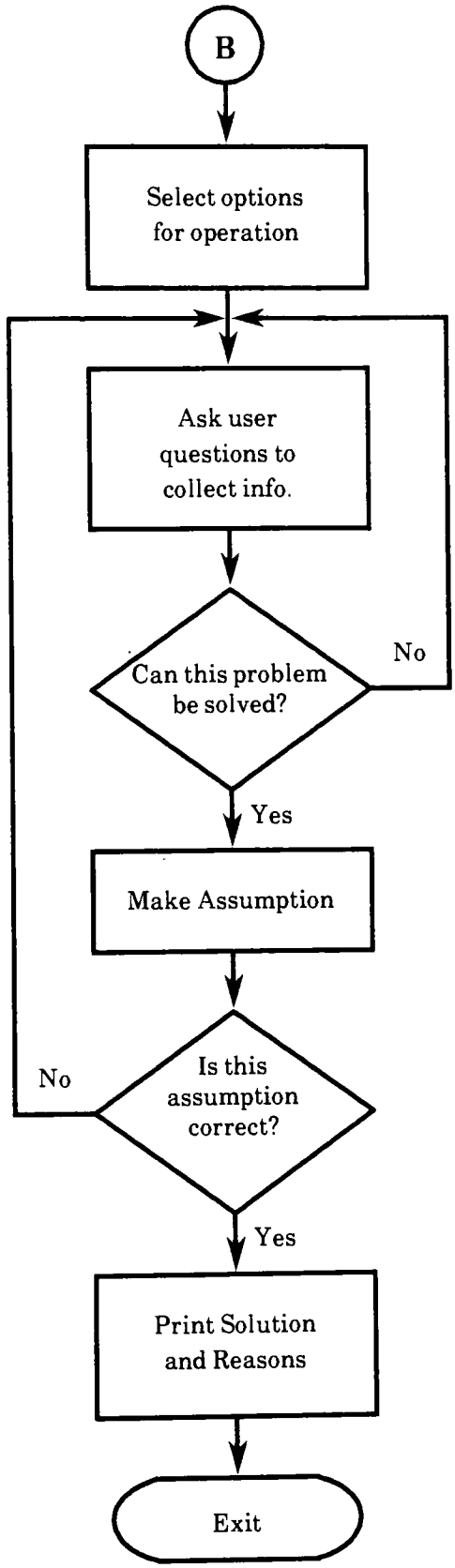
Appendix D -- AMUSED Flow Charts



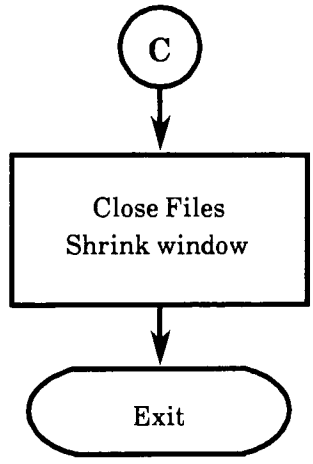
AMUSED Basic Operations
D.1



Assistance Mode
D.2

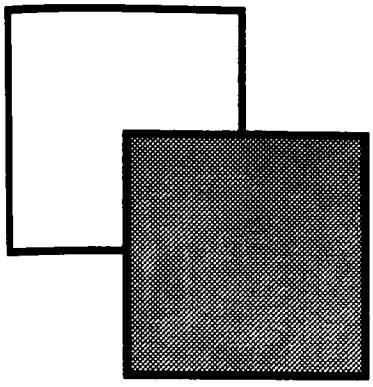


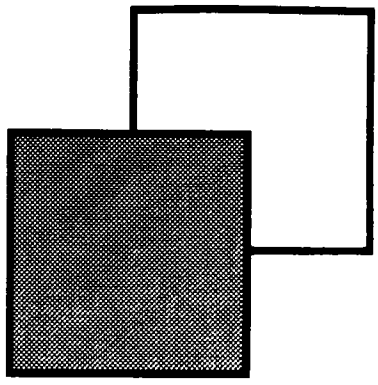
Diagnose Mode
D.3



Done
D.4

Appendix E -- AMUSED Source Code





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-CONTEXT.;6

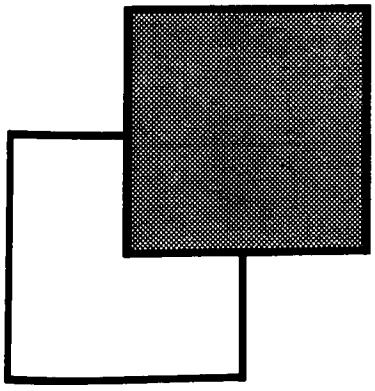
Created 1-May-89 8:40:56

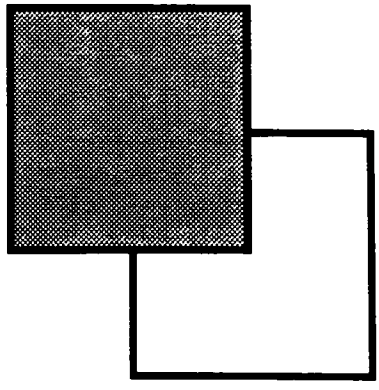
Printed 1-May-89 8:42:57

1 Sheet(s), 1 Copy.

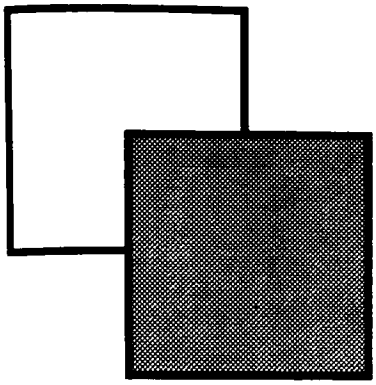
Xerox Print Service 10.0 on Palette

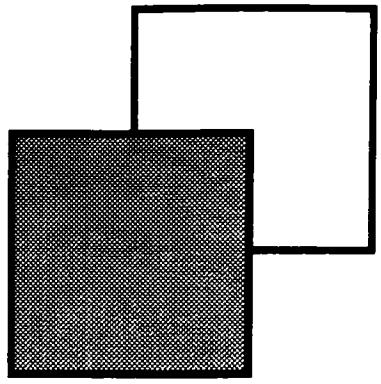
Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(DEFINE-FILE-INFO *PACKAGE "XCL-USER" *READTABLE "XCL" *BASE 10)
(IL:FILECREATED "27-Feb-89 16:10:22" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-CONTEXT.\;6 2175

      IL:|changes| IL:|to:|  (IL:FNS DEFINE.CONTEXT.BASE)

      IL:|previous| IL:|date:| "27-Feb-89 16:02:50" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-CONTEXT.\;5
)


(IL:PRETTYCOMPRINT IL:AMUSED-CONTEXTCOMS)

(IL:RPAQQ IL:AMUSED-CONTEXTCOMS ((IL:P (IN-PACKAGE 'IL:XCL-USER))
                                 (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                                        IL:AMUSED-RULES)
                                 (IL:FNS DEFINE.CONTEXT.BASE)
                                 (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY
                                        IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                    (IL:NLAML)
                                                                    (IL:LAMA DEFINE.CONTEXT.BASE)))))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:AMUSED-RULES IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL" :BASE
                                                                          10))
(IL:DEFINEQ

(DEFINE.CONTEXT.BASE
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 27-Feb-89 16:00 by MAFoltman")
          (BLOCK DEFINE.CONTEXT.BASE
           (IL:* IL:|;;| "Defines the basic context of local conditions used by this expert system.")
           (IL:* IL:|;;| "")

                 (DEFCONTEXT 'ERROR '((CRITICAL NIL)
                                      (FILENAME ATOM "What is the name of the file being used?")
                                      (LINKFILE ATOM "What is the name of the linkfile used?")
                                      (CAUSE "The probable cause")
                                      (SOLUTION "The solution you think it is")) '(CRITICAL)
                        '(SOLUTION)))))
)
(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA DEFINE.CONTEXT.BASE)
)
(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL (1203 1961 (DEFINE.CONTEXT.BASE 1216 . 1959)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-FREEMENU.;2

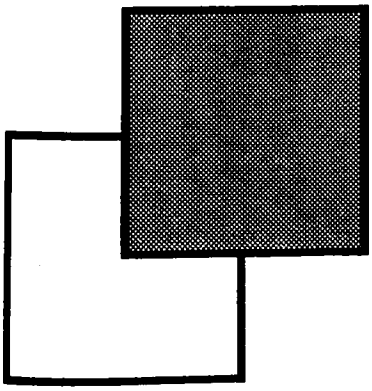
Created 1-May-89 8:42:57

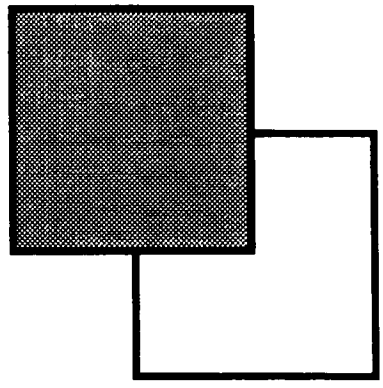
Printed 1-May-89 8:43:08

1 Sheet(s), 1 Copy.

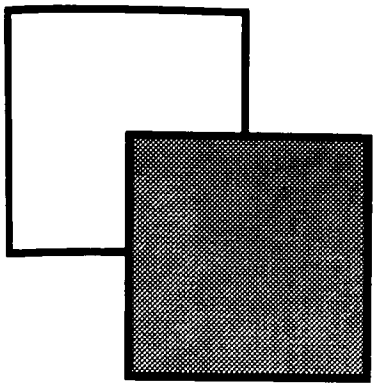
Xerox Print Service 10.0 on Palette

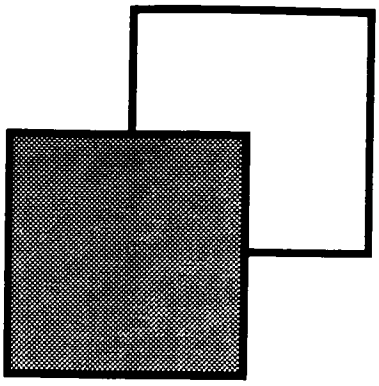
Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(DEFINE-FILE-INFO #PACKAGE "XCL-USER" #READTABLE "XCL" #BASE 10)
(IL:FILECREATED "27-Feb-89 16:10:53" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-FREEMENU.\;2 3777

      IL:|changes| IL:|to:|  (IL:FNS AM.DIAGNOSE.FREEMENU)

      IL:|previous| IL:|date:| " 6-Feb-89 09:11:35" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-FREEMENU.\;1
)


(IL:PRETTYCOMPRINT IL:AMUSED-FREEMENUCOMS)

(IL:RPAQQ IL:AMUSED-FREEMENUCOMS ((IL:P (IN-PACKAGE 'IL:XCL-USER))
                                  (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                                        IL:AMUSED-FREEMENU)
                                  (IL:FNS AM.DIAGNOSE.FREEMENU)
                                  (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY
                                        IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                    (IL:NLAML)
                                                                    (IL:LAMA AM.DIAGNOSE.FREEMENU)))))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:AMUSED-FREEMENU IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL"
                                                                :BASE 10))
(IL:DEFINEQ

(AM.DIAGNOSE.FREEMENU
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 27-Feb-89 16:07 by MAFoltman")
          (BLOCK AM.DIAGNOSE.FREEMENU
           (IL:* IL:|;;| "Set up the option window for diagnose global options.")
           (IL:* IL:|;;| "")

                (SETQ DIAGNOSE.OPTIONS
                      (IL:FREEMENU '(((IL:GROUP IL:ID ROW2
                                              ((IL:LABEL "OPTIONS:" IL:FONT (MODERN 12 BOLD))
                                               (TYPE IL:TOGGLE IL:LABEL ENABLE-LOG IL:COLLECTION ROW2
                                                     IL:ID ENABLELOG)
                                               (IL:GROUP (IL:PROPS IL:ID ROW2B IL:LEFT 100 IL:FONT
                                                                   (MODERN 12 BOLD))
                                                      ((IL:LABEL COMPLETE! IL:SELECTEDFN
                                                              AM.SET.GLOBAL.OPTIONS))))))
                                      ((IL:LABEL
                                            "-------------------------------------------------------"
                                            ))
                                      ((IL:GROUP IL:ID ROW3 ((IL:LABEL IL:TOOL IL:FONT
                                                                    (MODERN 12 BOLD))
                                                             (TYPE IL:NWAY IL:LABEL LINK IL:COLLECTION
                                                                   ROW3 IL:NWAYPROPS
                                                                   (DESELECT T IL:ID IL:TOOL))
                                                             (TYPE IL:NWAY IL:LABEL LINK8051
                                                                   IL:COLLECTION ROW3)))
                                       (IL:GROUP (IL:PROPS IL:ID ROW3B IL:LEFT 50)
                                              ((IL:LABEL IL:PROCESSOR IL:FONT (MODERN 12 BOLD))
                                               (TYPE IL:NWAY IL:LABEL 8085 IL:COLLECTION ROW3B
                                                     IL:NWAYPROPS (DESELECT T IL:ID IL:PROCESSOR))
                                               (TYPE IL:NWAY IL:LABEL 8051 IL:COLLECTION ROW3B))))))))
          ))
)
(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA AM.DIAGNOSE.FREEMENU)
)
(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL (1226 3562 (AM.DIAGNOSE.FREEMENU 1239 . 3560)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-QUESTION-DATABASE.;8

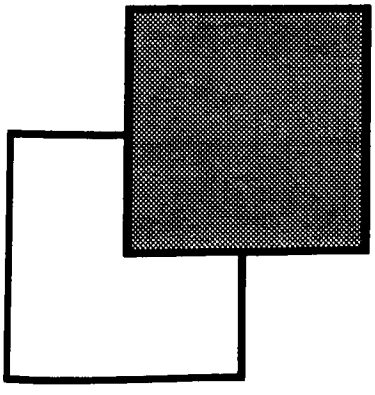
Created 1-May-89 8:43:06

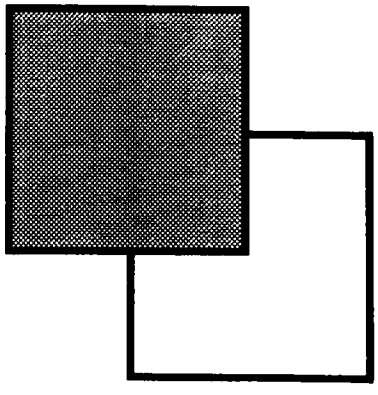
Printed 1-May-89 8:43:23

3 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





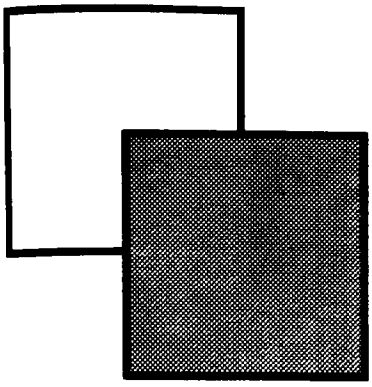
```
(DEFINE-FILE-INFO ▮PACKAGE "XCL-USER" ▮READTABLE "XCL" ▮BASE 10)
(IL:FILECREATEO "13-Mar-89 16:31:37" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-QUESTION-DATABASE.\;8
 11028

      IL:|changes| IL:|to:|  (IL:FNS DEFINE.LINK8051.QUESTIONS DEFINE.LINK.QUESTIONS)

      IL:|previous| IL:|date:| " 2-Mar-89 16:24:14"
IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-QUESTION-DATABASE.\;6)


(IL:PRETTYCOMPRINT IL:AMUSED-QUESTION-DATABASECOMS)

(IL:RPAQQ IL:AMUSED-QUESTION-DATABASECOMS ((IL:P (IN-PACKAGE 'IL:XCL-USER))
                                           (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                                                  IL:AMUSED-QUESTION-DATABASE)
                                           (IL:FNS DEFINE.LINK.QUESTIONS
                                                  DEFINE.LINK8051.QUESTIONS)
                                           (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE
                                                  IL:DONTCOPY IL:COMPILERVARS
                                                  (IL:ADDVARS (IL:NLAMA)
                                                         (IL:NLAML)
                                                         (IL:LAMA DEFINE.LINK8051.QUESTIONS
                                                                DEFINE.LINK.QUESTIONS)))))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:AMUSED-QUESTION-DATABASE IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE
                                                                          "XCL" :BASE 10))
(IL:DEFINEQ

(DEFINE.LINK.QUESTIONS
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 28-Feb-89 09:22 by MAFoltman")
        (BLOCK DEFINE.LINK.QUESTIONS
          (IL:* IL:|;;|
        "Database of questions based on the Link tool. The questions are separated ")
          (IL:* IL:|;;| "into categories for easy retrieval and organization.")
          (IL:* IL:|;;| "")

              (SETQ LINK.QUESTIONS '((CONTENT ((1 "Is there a problem with the command line?")
                                               (2
                                                "Are there invalid characters in the command line?"
                                                  )
                                               (3 "Are the characters typed in by mistake?")))
                                     (FILE ((1 "Is there a file problem?")
                                            (2 "Can the file be opened?")
                                            (3 "Is it the target file that cannot be opened?")
                                            (4
                                        "Is the file name listed with a directory in the User.cm?"
                                              )))
                                     (SPACE ((1 "Is there a space allocation problem?")
                                             (2
                                               "Are the STCB ranges too small for a given section?"
                                               )
                                             (3
                                           "Are the modules declared in the correct storage class?"
                                               )
                                             (4
                                    "Is the default code class set to the application code section?"
                                              )))
                                     (OPTION NIL)
                                     (MISC NIL)
                                     (WARN ((1 "Is there a referencing problem with the files?")
                                            (2 "Was the file accessed in more than one version?")
                                            (3
                          "Were the files using the uncommon version assembled with the newer version?"
                                               )
                                            (4
    "Is this latest file located on the link workstation and is the file on the current search path?"
                                               )
                                            (5 "Is there a byte definition problem?")
                                            (6 "Is it a multiple definition problem?")
                                            (7 "Was the correct STD file used for the link?")
                                            (8 "Is the reference undefined?")
                                            (9
```

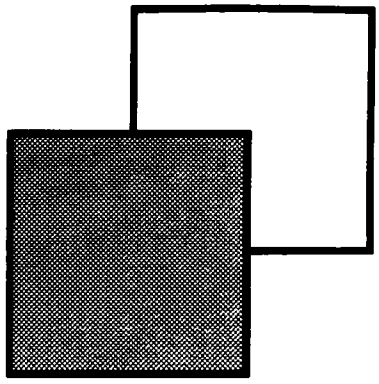
```
                "Was the wrong version of the specified include file used in the modules mentioned?"
                                        )
                                    (10 "Is there a file definition problem?")
                                    (11 "Is the variable mutliply defined?")
                                    (12
                                "Are there storage class problems in the PSIZE information?"
                                        )
                                    (13
                                  "Are the names missing from the storage class name table?"
                                        )))
                                (FATAL NIL))))))
```

(DEFINE.LINK8051.QUESTIONS

```
  (LAMBOA NIL                                   (IL:* IL:\; "Edited 2-Mar-89 16:24 by MAFoltman")
          (BLOCK OEFINE.LINKB051.QUESTIONS
           (IL:* IL:|;;| "Database of questions based on the Link8051 tool. The questions are ")
           (IL:* IL:|;;| "separated into categories for easy retrieval and organization.")
           (IL:* IL:|;;| "")

                (SETQ LINK8051.QUESTIONS '((FILE ((1 "Is it a file problem?")
                                                  (2 "Is the file name correct?")
                                                  (3 "Is the file on the Link workstation?")
                                                  (4
                                             "Is the file on the workstation's current search path?"
                                                     )
                                                  (5 "Is the file an invalid 8051 Format ROM file?"))
                                                 )
                                           (CONTENT ((1
                                              "Is there a problem with the content of the link file?"
                                                        )
                                                     (2 "Is the problem with the checksums?")
                                                     (3 "Oo the ranges overlap?")
                                                     (4
                                                     "Was a premature end of file (EOF) encountered?"
                                                        )
                                                     (5
                                               "Is the 'ENO LINK' missing at the end of the .link file?"
                                                        )
                                                     (6 "Are the data segments incorrect?")
                                                     (7
                                          "Ooes the range overlap with a previously defined data segment"
                                                        )
                                                     (8 "Is the other data segment in a ROM file?")))
                                           (OPTION ((1
                                                     "Is there a problem with the options specified?"
                                                      )
                                                    (2
                                              "Are all of the options specified in this list -- a-x?"
                                                      )
                                                    (3 "Was an incorrect switch used?")
                                                    (4
                                                 "Is there a valid combination of options specified?"
                                                       )))
                                           (SPACE ((1 "Is there a space allocation error?")
                                                   (2
                                                     "Is there no room available for the allocation?"
                                                     )
                                                   (3 "Is the allocation for a data segment?")
                                                   (4
                                                  "Is the allocation overwritten by another segment?"
                                                      )))
                                           (MISC NIL)
                                           (WARN ((1 "Is there a referencing problem with the files?"
                                                     )
                                                  (2
                                                    "Was the file accessed in more than one version?"
                                                     )
                                                  (3
                        "Were the files using the uncommon versions assembled with the newer version?"
                                                     )
                                                  (4
    "Is this latest file located on the link workstation and is the file on the current search path?"
                                                     )
                                                  (5 "Is there a definition problem in the file?")
```

```
                                                    (6 "Is the definition missing?")
                                                    (7
                                                     "Is the missing definition a relocatable segment?"
                                                       )
                                                    (8 "Is there a space allocation problem?")
                                                    (9
                                                    "Is it an absolute data segment that is too large?"
                                                       )
                                                    (10 "Will the segment possibly be truncated?")))
                                               (FATAL NIL))))))
)
(IL:OECLARE\: IL:OONTEVAL@LOAO IL:OOEVAL@COMPILE IL:OONTCOPY IL:COMPILERVARS

(IL:AOOTOVAR IL:NLAMA )

(IL:AOOTOVAR IL:NLAML )

(IL:AOOTOVAR IL:LAMA OEFINE.LINK8051.QUESTIONS OEFINE.LINK.QUESTIONS)
)
(IL:OECLARE\: IL:OONTCOPY
  (IL:FILEMAP (NIL (1579 10786 (OEFINE.LINK.QUESTIONS 1592 . 5471) (OEFINE.LINK8051.QUESTIONS 5473 .
10784)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-RULES.;17

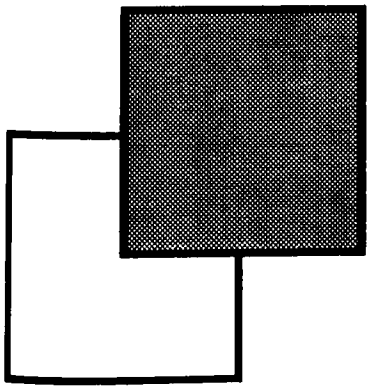
Created 1-May-89 8:43:27

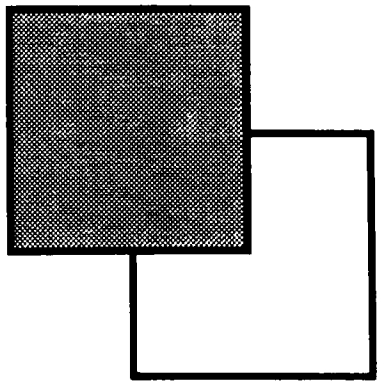
Printed 1-May-89 8:44:24

7 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(DEFINE-FILE-INFO ▮PACKAGE "XCL-USER" ▮READTABLE "XCL" ▮BASE 10)
(IL:FILECREATED "14-Mar-89 14:41:36" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-RULES.\;17 30599

      IL:|changes| IL:|to:|  (IL:FNS DEFINE.RULE.BASE.WARNING DEFINE.RULE.BASE.FATAL
                                    DEFINE.RULE.BASE.8051.FATAL DEFINE.RULE.BASE.8051.WARNING
                                    DEFINE.RULE.BASE.8051.ERROR DEFINE.RULE.BASE.ERROR)

      IL:|previous| IL:|date:| " 6-Mar-89 17:46:56" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-RULES.\;14
)


(IL:PRETTYCOMPRINT IL:AMUSED-RULESCOMS)

(IL:RPAQQ IL:AMUSED-RULESCOMS ((IL:P (IN-PACKAGE 'IL:XCL-USER))
                               (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                                      IL:AMUSED-RULES)

        (IL:* IL:|;;| "Rule bases for Link errors.")

                               (IL:FNS DEFINE.RULE.BASE.WARNING DEFINE.RULE.BASE.ERROR
                                      DEFINE.RULE.BASE.FATAL)

        (IL:* IL:|;;| "Rule bases for Link8051 errors")

                               (IL:FNS DEFINE.RULE.BASE.8051.WARNING DEFINE.RULE.BASE.8051.ERROR
                                      DEFINE.RULE.BASE.8051.FATAL)
                               (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY
                                      IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                  (IL:NLAML)
                                                                  (IL:LAMA DEFINE.RULE.BASE.8051.FATAL
                                                                         DEFINE.RULE.BASE.8051.ERROR
                                                                         DEFINE.RULE.BASE.8051.WARNING
                                                                         DEFINE.RULE.BASE.FATAL
                                                                         DEFINE.RULE.BASE.ERROR
                                                                         DEFINE.RULE.BASE.WARNING)))))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:AMUSED-RULES IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL" :BASE
                                                           10))


(IL:* IL:|;;| "Rule bases for Link errors.")

(IL:DEFINEQ

(DEFINE.RULE.BASE.WARNING
  (LAMBDA NIL                                   (IL:* IL:\; "Edited 6-Mar-89 17:46 by MAFoltman")
        (BLOCK DEFINE.RULE.BASE.WARNING
         (IL:* IL:|;;| "This set of rules is specific to the Link tool and those error messages")
         (IL:* IL:|;;| "that are classified as warnings. The rules are a set of conditions to")
         (IL:* IL:|;;| " determine the exact cause of each of the messages encountered.")
         (IL:* IL:|;;| "")

               (DEFRULES (RULE001 ($AND (NOTSAME CNTXT CRITICAL YES)
                                        (SAME.GLOBAL FOUND? NIL)
                                        (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                        (SAME.GLOBAL LINKER.USED 'LINK)
                                        (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 1)
                                               'YES)
                                        (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 2)
                                               'YES)
                                        (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 3)
                                               'NO))
                                   (DO-ALL (PRINT.STRING
                                     "Retrieve the newer version of the file, and compile against it."
                                                )
                                           (SETQ FOUND? T)
                                           (CONCLUDE CNTXT SOLUTION "Version mismatch" TALLY 900)))
                         (RULE002 ($AND (NOTSAME CNTXT CRITICAL YES)
                                        (SAME.GLOBAL FOUND? NIL)
                                        (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                        (SAME.GLOBAL LINKER.USED 'LINK)
                                        (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 1)
```

```
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 2)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 3)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 4)
                                        'YES))
                          (OO-ALL (SETQ FOUNO? T)
                                 (PRINT.STRING
                    "Reset the search path, or move the file to a directory on the active search path."
                                 )
                                 (CONCLUOE CNTXT SOLUTION "Version mismatch" TALLY 900)))
                    (RULE003 ($ANO (NOTSAME CNTXT CRITICAL YES)
                                 (SAME.GLOBAL FOUNO? NIL)
                                 (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                 (SAME.GLOBAL LINKER.USEO 'LINK)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 5)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 6)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 7)
                                        'NO))
                          (OO-ALL (SETQ FOUNO? T)
                                 (PRINT.STRING "Load the correct .std file to the search path.")
                                 (CONCLUOE CNTXT SOLUTION "Wrong STO used" TALLY 900)))
                    (RULE004 ($ANO (NOTSAME CNTXT CRITICAL YES)
                                 (SAME.GLOBAL FOUNO? NIL)
                                 (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                 (SAME.GLOBAL LINKER.USEO 'LINK)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 1)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 8)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 9)
                                        'YES))
                          (OO-ALL (SETQ FOUNO T)
                                 (PRINT.STRING
                          "Recompile those modules with the correct version of the include file."
                                 )
                                 (CONCLUOE CNTXT SOLUTION "Undefined reference" TALLY 900)))
                    (RULE005 ($ANO (NOTSAME CNTXT CRITICAL YES)
                                 (SAME.GLOBAL FOUNO? NIL)
                                 (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                 (SAME.GLOBAL LINKER.USEO 'LINK)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 10)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 6)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 11)
                                        'YES))
                          (OO-ALL (SETQ FOUNO T)
                                 (PRINT.STRING "Oetermine which definition is correct.")
                                 (PRINT.STRING
                                       "Adjust the other file to reference the correct variable."
                                       )
                                 (CONCLUOE CNTXT SOLUTION "Multiple Oefinition" TALLY 900)))
                    (RULE006 ($ANO (NOTSAME CNTXT CRITICAL YES)
                                 (SAME.GLOBAL FOUNO? NIL)
                                 (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                 (SAME.GLOBAL LINKER.USEO 'LINK)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 12)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 13)
                                        'YES))
                          (OO-ALL (SETQ FOUNO T)
                                 (PRINT.STRING "These names are currently not supported.")
                                 (PRINT.STRING
                                        "Move the objects listed to other storage classes.")
                                 (CONCLUOE CNTXT SOLUTION "Bad Storage Class Names" TALLY 900))))
              )))

(DEFINE.RULE.BASE.ERROR
  (LAM80A NIL                                    (IL:* IL:\; "Edited 1-Mar-89 15:48 by MAFoltman")
        (BLOCK OEFINE.RULE.BASE.ERROR
```

```
    (IL:* IL:|;;|
"This set of rules is specific to the Link tool and that set of error messages ")
    (IL:* IL:|;;|
" that are classified as errors in the message. Each rule is a set of conditions which")
    (IL:* IL:|;;| " must be true in order for a specific cause of the error to be true.")
    (IL:* IL:|;;| "")

        (DEFRULES (RULE001 ($AND ($OR (NOTSAME CNTXT CRITICAL YES)
                                      (SAME CNTXT CRITICAL YES))
                                 (SAME.GLOBAL FOUND? NIL)
                                 (SAME.GLOBAL ERROR.TYPE 'ERROR)
                                 (SAME.GLOBAL LINKER.USED 'LINK)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 1)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 2)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 3)
                                        'YES)
                                 (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 4)
                                        'YES))
                           (DO-ALL (SETQ FOUND? T)
                                  (PRINT.STRING "Please check the section codes to make sure")
                                  (PRINT.STRING "it matches the application section number.")
                                  (PRINT.STRING "If the codes are set within assembly code,")
                                  (PRINT.STRING "an incorrect number assignment could be made."
                                         )
                                  (CONCLUDE CNTXT SOLUTION "Bad section code" TALLY 900)))
                (RULE002 ($AND ($OR (NOTSAME CNTXT CRITICAL YES)
                                    (SAME CNTXT CRITICAL YES))
                               (SAME.GLOBAL FOUND? NIL)
                               (SAME.GLOBAL ERROR.TYPE 'ERROR)
                               (SAME.GLOBAL LINKER.USED 'LINK)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 1)
                                      'YES)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 2)
                                      'NO)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 3)
                                      'YES)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 4)
                                      'YES))
                         (DO-ALL (SETQ FOUND? T)
                                (PRINT.STRING "Adjust the user.cm to have no directory")
                                (PRINT.STRING "on the target name in the target slot.")
                                (CONCLUDE CNTXT SOLUTION "Bad target directory" TALLY 900)))
                (RULE003 ($AND ($OR (NOTSAME CNTXT CRITICAL YES)
                                    (SAME CNTXT CRITICAL YES))
                               (SAME.GLOBAL FOUND? NIL)
                               (SAME.GLOBAL ERROR.TYPE 'ERROR)
                               (SAME.GLOBAL LINKER.USED 'LINK)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 1)
                                      'YES)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 2)
                                      'YES)
                               (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 3)
                                      'YES))
                         (DO-ALL (SETQ FOUND? T)
                                (PRINT.STRING
                                      "Eliminate these bad characters and try the run again.")
                                (CONCLUDE CNTXT SOLUTION "Illegal characters in command line"
                                      TALLY 900)))))))

(DEFINE.RULE.BASE.FATAL
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 1-Mar-89 15:47 by MAFoltman")
        (BLOCK DEFINE.RULE.BASE.FATAL
          (IL:* IL:|;;| "This set of rules is specific to the Link tool and those error messages ")
          (IL:* IL:|;;|
"that are classified as FATAL. Each rule represents a given set of conditions")
          (IL:* IL:|;;|
" and questions which must be true in order for the cause to be correct for the")
          (IL:* IL:|;;| "given error message.")
          (IL:* IL:|;;| "")

        (DEFRULES (RULE001 ($AND (SAME CNTXT CRITICAL YES)
                                 (SAME.GLOBAL FOUND? NIL)
```

```
                                        (SAME.GLOBAL ERROR.TYPE 'FATAL)
                                        (SAME.GLOBAL LINKER.USED 'LINK))
                                  (DO-ALL (SETQ FOUND T)
                                          (CONCLUDE CNTXT SOLUTION 'FATAL TALLY 900)))))))
)
```

(IL:* IL:|;;| "Rule bases for Link8051 errors")

```
(IL:DEFINEQ
```

(DEFINE.RULE.BASE.8051.WARNING

```
  (LAMBDA NIL                                   (IL:* IL:\; "Edited 1-Mar-89 15:46 by MAFoltman")
        (BLOCK DEFINE.RULE.BASE.8051.WARNING
         (IL:* IL:|;;| "This set of rules is specific to the LINK8051 tool and those ")
         (IL:* IL:|;;| "error messages that are classified as warnings.  Each rule contains a")
         (IL:* IL:|;;| " set of conditions and questions which must be true in order to ")
         (IL:* IL:|;;| " correctly identify the resultant cause of the problem.")
         (IL:* IL:|;;| "")

              (DEFRULES (RULE001 ($AND (NOTSAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUND? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 1)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 2)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 3)
                                              'NO))
                                 (DO-ALL (SETQ FOUND? T)
                                         (PRINT.STRING
                                   "Retrieve the newer version of the file, and assemble against it."
                                               )
                                         (CONCLUDE CNTXT SOLUTION "Version mismatch" TALLY 900)))
                        (RULE002 ($AND (NOTSAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUNO? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 1)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 2)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 3)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 4)
                                              'YES))
                                  (OO-ALL (SETQ FOUND? T)
                                          (PRINT.STRING
      "Reset the search path, or move the file to a different directory on the active search path."
                                                )
                                          (CONCLUDE CNTXT SOLUTION "Version mismatch" TALLY 900)))
                        (RULE003 ($AND (NOTSAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUNO? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 5)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 6)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 7)
                                              'YES))
                                  (DO-ALL (SETQ FOUNO? T)
                                          (PRINT.STRING
                                            "Make sure the segments are not declared at the module level."
                                                )
                                          (PRINT.STRING
                                                 "The segment is already definded within the module.")
                                          (CONCLUDE CNTXT SOLUTION "Undefined segment" TALLY 900)))
                        (RULE004 ($ANO (NOTSAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUNO? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'WARNING)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
```

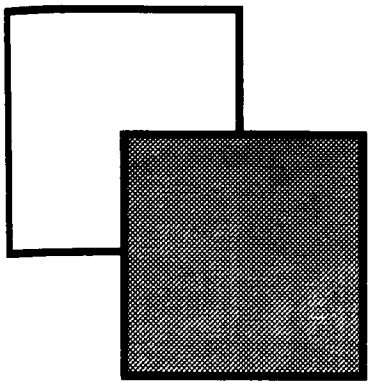
```
                                    (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 8)
                                           'YES)
                                    (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 9)
                                           'YES)
                                    (SAME.GLOBAL (SPECIFIC.QUESTION 'WARN 10)
                                           'YES))
                             (DO-ALL (SETQ FOUND? T)
                                    (PRINT.STRING "The DSEG range appears to be full.")
                                    (PRINT.STRING "Please check the range of elements within this")
                                    (PRINT.STRING "segment. If the warning persists, omit these")
                                    (PRINT.STRING "segments and report the problem to your Darwin")
                                    (PRINT.STRING "representative.")
                                    (CONCLUDE CNTXT SOLUTION "Full DSEG range" TALLY 800)))))))
```

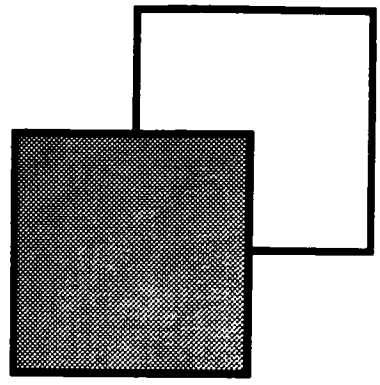
(DEFINE.RULE.BASE.8051.ERROR

```
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 2-Mar-89 16:21 by MAFoltman")
        (BLOCK DEFINE.RULE.BASE.8051.ERROR
         (IL:* IL:|;;| "This set of rules is specific to the LINK8051 tool and those")
         (IL:* IL:|;;| " error messages which are classified as actual ERRORs on the")
         (IL:* IL:|;;| " message. Each rule contains a set of conditions and specific")
         (IL:* IL:|;;| " questions which are required to be true in order for the ")
         (IL:* IL:|;;| " actual cause of the problem to be found.")
         (IL:* IL:|;;| "")

              (DEFRULES (RULE001 ($AND (SAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUND? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'ERROR)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 1)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 2)
                                              'NO)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 3)
                                              'YES))
                                  (DO-ALL (SETQ FOUND? T)
                                         (PRINT.STRING
                                                "Choose the correct switch to eliminate this problem."
                                                )
                                         (CONCLUDE CNTXT SOLUTION "incorrect switch" TALLY 900)))
                        (RULE002 ($AND (SAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUND? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'ERROR)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 1)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 2)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'OPTION 4)
                                              'NO))
                                  (DO-ALL (SETQ FOUND? T)
                                         (PRINT.STRING
                                        "Check the combination of switches specified in the Link manual."
                                                )
                                         (CONCLUDE CNTXT SOLUTION "Incorrect combination" TALLY 900)))
                        (RULE003 ($AND (SAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUND? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'ERROR)
                                       (SAME.GLOBAL LINKER.USED 'LINK8051)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 1)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 2)
                                              'YES)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 3)
                                              'NO)
                                       (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 4)
                                              'YES))
                                  (DO-ALL (SETQ FOUND? T)
                                         (PRINT.STRING
    "Correct the workstation search path, or move the file to a directory on the active search path."
                                                )
                                         (CONCLUDE CNTXT SOLUTION "Missing file" TALLY 900)))
                        (RULE004 ($AND (SAME CNTXT CRITICAL YES)
                                       (SAME.GLOBAL FOUND? NIL)
                                       (SAME.GLOBAL ERROR.TYPE 'ERROR)
```

```
                        (SAME.GLOBAL LINKER.USEO 'LINK8051)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 1)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 2)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 3)
                              'YES))
                  (OO-ALL (SETQ FOUNO? T)
                        (PRINT.STRING
                              "Check the checksum ranges for overlapping values.")
                        (CONCLUOE CNTXT SOLUTION "Checksum overlap" TALLY 900)))
            (RULE005 ($ANO (SAME CNTXT CRITICAL YES)
                        (SAME.GLOBAL FOUNO? NIL)
                        (SAME.GLOBAL ERROR.TYPE 'ERROR)
                        (SAME.GLOBAL LINKER.USEO 'LINK8051)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 1)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 4)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 5)
                              'YES))
                  (OO-ALL (SETQ FOUNO? T)
                        (PRINT.STRING
                              "Add the ENO LINK statement to the end of the file.")
                        (CONCLUOE CNTXT SOLUTION "EOF encountered" TALLY 900)))
            (RULE006 ($ANO (SAME CNTXT CRITICAL YES)
                        (SAME.GLOBAL FOUNO? NIL)
                        (SAME.GLOBAL ERROR.TYPE 'ERROR)
                        (SAME.GLOBAL LINKER.USEO 'LINK8051)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 1)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'FILE 5)
                              'YES))
                  (OO-ALL (SETQ FOUNO? T)
                        (PRINT.STRING "Check to make sure that a working version ")
                        (PRINT.STRING "of LINK8051 is being used.")
                        (CONCLUOE CNTXT SOLUTION "Invalid ROM File" TALLY 900)))
            (RULE007 ($ANO (SAME CNTXT CRITICAL YES)
                        (SAME.GLOBAL FOUNO? NIL)
                        (SAME.GLOBAL ERROR.TYPE 'ERROR)
                        (SAME.GLOBAL LINKER.USEO 'LINK8051)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 1)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 2)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 3)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'SPACE 4)
                              'YES))
                  (OO-ALL (SETQ FOUNO? T)
                        (PRINT.STRING "Check the allocation sizes and locations")
                        (PRINT.STRING "of all segments for possible overlaps.")
                        (CONCLUOE CNTXT SOLUTION "segment overlap" TALLY 900)))
            (RULE00B ($ANO ($OR (SAME CNTXT CRITICAL YES)
                              (NOTSAME CNTXT CRITICAL YES))
                        (SAME.GLOBAL FOUNO? NIL)
                        (SAME.GLOBAL ERROR.TYPE 'ERROR)
                        (SAME.GLOBAL LINKER.USEO 'LINK8051)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 1)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 6)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 7)
                              'YES)
                        (SAME.GLOBAL (SPECIFIC.QUESTION 'CONTENT 8)
                              'YES))
                  (OO-ALL (SETQ FOUNO? T)
                        (PRINT.STRING "The OSEG range appears to be full.")
                        (PRINT.STRING "Please check the range of elements within")
                        (PRINT.STRING "this segment.  If the warning persists, ")
                        (PRINT.STRING "omit these segments and report the problem")
                        (PRINT.STRING "to the Oarwin Representative.")
                        (CONCLUOE CNTXT SOLUTION "Full OSEG range" TALLY 900)))))))
```

```
(DEFINE.RULE.BASE.8051.FATAL
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 1-Mar-89 15:44 by MAFoltman")
        (BLOCK DEFINE.RULE.BASE.8051.FATAL
         (IL:* IL:;;; " This set of rules is specific to the LINK8051 tool and those error ")
         (IL:* IL:;;; "messages that are classified as FATAL. Each rule contains a set of ")
         (IL:* IL:;;; " conditions and specific questions which must be answered ")
         (IL:* IL:;;; " correctly in order for the actual cause of the message to be found.")
         (IL:* IL:;;; "")

               (DEFRULES (RULE001 ($AND (SAME CNTXT CRITICAL YES)
                                        (SAME.GLOBAL FOUND? NIL)
                                        (SAME.GLOBAL ERROR.TYPE 'FATAL)
                                        (SAME.GLOBAL LINKER.USED 'LINK8051))
                                  (DO-ALL (SETQ FOUND? T)
                                         (CONCLUDE CNTXT SOLUTION 'FATAL8051 TALLY 900)))))))
)
(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA DEFINE.RULE.BASE.8051.FATAL DEFINE.RULE.BASE.8051.ERROR
                              DEFINE.RULE.BASE.8051.WARNING DEFINE.RULE.BASE.FATAL
                              DEFINE.RULE.BASE.ERROR DEFINE.RULE.BASE.WARNING)
)
(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL (2411 14539 (DEFINE.RULE.BASE.WARNING 2424 . 9111) (DEFINE.RULE.BASE.ERROR 9113
13504) (DEFINE.RULE.BASE.FATAL 13506 . 14537)) (14596 30184 (DEFINE.RULE.BASE.8051.WARNING 14609 .
19852) (DEFINE.RULE.BASE.8051.ERROR 19854 . 29141) (DEFINE.RULE.BASE.8051.FATAL 29143 . 30182)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-TOOLS.;26

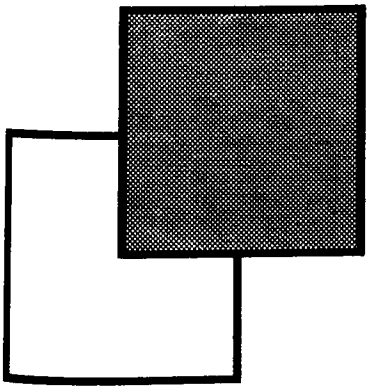
Created 1-May-89 8:44:31

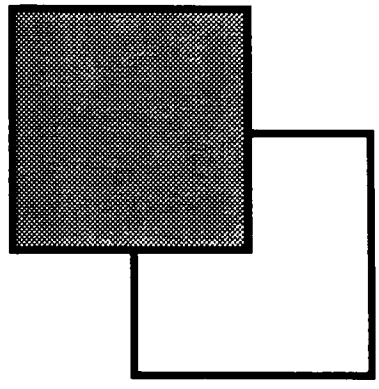
Printed 1-May-89 8:45:13

6 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(DEFINE-FILE-INFO ▮PACKAGE "XCL-USER" ▮READTABLE "XCL" ▮BASE 10)
(IL:FILECREATEO " 3-Apr-89 14:39:02" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-TOOLS.\;26 19898

      IL:|changes| IL:|to:|  (IL:FNS PRINT.STRING PRINC.STRING AM.NEW.LINE CREATE.LOG.FILE
                                    COLLECT.USER.CERTAINTY)

      IL:|previous| IL:|date:| "17-Mar-89 15:57:36" IL:{OSK}<LISPFILES>THESIS>COOE>AMUSEO-TOOLS.\;25
)


(IL:PRETTYCOMPRINT IL:AMUSEO-TOOLSCOMS)

(IL:RPAQQ IL:AMUSED-TOOLSCOMS
          ((IL:P (IN-PACKAGE 'XCL-USER))
           (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                  IL:AMUSEO-TOOLS)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| "Basic I/O functions")

           (IL:FNS PRINT.STRING PRINC.STRING AM.NEW.LINE AM.GET.INPUT AM.GET.INPUT.NUMBER
                  PRINT.QUESTION AM.ERROR)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| "String manipulations")

           (IL:FNS REVERSE.STRING AM.ISOLATE.KEY.TEXT)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| "Functions for Global variable checking inside of rules")

           (IL:FNS SAME.GLOBAL SPECIFIC.QUESTION)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;|
        " Functions for seting the global variables from the state of the freemenu options.")

           (IL:FNS AM.SET.GLOBAL.OPTIONS)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| " User Interface functions")

           (IL:FNS CREATE.LOG.FILE CREATE.MENU COLLECT.USER.CERTAINTY)
           (IL:OECLARE\: IL:OONTEVAL@LDAO IL:OOEVAL@COMPILE IL:OONTCOPY IL:COMPILERVARS
                  (IL:AOOVARS (IL:NLAMA)
                         (IL:NLAML)
                         (IL:LAMA COLLECT.USER.CERTAINTY CREATE.MENU AM.SET.GLOBAL.OPTIONS
                                SPECIFIC.QUESTION SAME.GLOBAL AM.ISOLATE.KEY.TEXT REVERSE.STRING
                                AM.ERROR PRINT.QUESTION AM.GET.INPUT.NUMBER AM.GET.INPUT AM.NEW.LINE
                                PRINC.STRING PRINT.STRING)))))
(IN-PACKAGE 'XCL-USER)

(IL:PUTPROPS IL:AMUSED-TOOLS IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL" :BASE
                                                                  10))
```

(IL:* IL:|;;| "")

(IL:* IL:|;;| "Basic I/O functions")

```
(IL:DEFINEQ

(PRINT.STRING
  (LAMBDA (STRING)                                    (IL:* IL:\; "Edited 3-Apr-89 14:30 by MAFoltman")
          (IL:* IL:|;;| " Print a string to the main Amused interaction window. If")
          (IL:* IL:|;;| " the log option is set, then print the same string to the Log File.")
          (IL:* IL:|;;| "")

          (PROG NIL
                (PRINC STRING AMUSED.MAIN.WINDOW)
                (TERPRI AMUSED.MAIN.WINDOW)
                (COND
                   (ENABLE.LOG (COND
                                  (TEMP.LOG.FILE (SETQ LOG.FILE (APPEND LOG.FILE
                                                                        (LIST (APPEND TEMP.LOG.FILE
                                                                                      (LIST STRING)))))
                                                 (SETQ TEMP.LOG.FILE NIL))
                                  (T (SETQ LOG.FILE (APPEND LOG.FILE (LIST (LIST STRING))))))))))))

(PRINC.STRING
  (LAMBDA (STRING)                                    (IL:* IL:\; "Edited 3-Apr-89 14:26 by MAFoltman")
          (BLOCK PRINC.STRING
           (IL:* IL:|;;| " Print a string to the main AMUSED window without printing")
           (IL:* IL:|;;| " a carriage return to the file. This feature will allow for more")
           (IL:* IL:|;;| " than one princ.string statement per line. If the global ")
           (IL:* IL:|;;| " log option is set, then print the string to the log file.")
           (IL:* IL:|;;| "")

                 (PROG NIL
                       (PRINC STRING AMUSED.MAIN.WINDOW)
                       (COND
                          (ENABLE.LOG (SETQ TEMP.LOG.FILE (APPEND TEMP.LOG.FILE (LIST STRING))))))))))

(AM.NEW.LINE
  (LAMBDA NIL                                         (IL:* IL:\; "Edited 3-Apr-89 14:33 by MAFoltman")
          (BLOCK AM.NEW.LINE
           (IL:* IL:|;;| " Print the end of line character to the main AMUSED window. If")
           (IL:* IL:|;;| " the global log option is set, then print to the log file also.")
           (IL:* IL:|;;| "")

                 (TERPRI AMUSED.MAIN.WINDOW)
                 (COND
                    (ENABLE.LOG (SETQ LOG.FILE (APPEND LOG.FILE (LIST TEMP.LOG.FILE)))
                           (SETQ TEMP.LOG.FILE NIL))))))))

(AM.GET.INPUT
  (LAMBDA (PROMPT.STRING CANDIDATE.STRING)            (IL:* IL:\; "Edited 8-Mar-89 07:57 by MAFoltman")
          (BLOCK AM.GET.INPUT
           (IL:* IL:|;;| " Retrieve input entered into the main AMUSED window.")
           (IL:* IL:|;;| " Returns a string.")
           (IL:* IL:|;;| "")

                 (PROG (VALUE)
                       (SETQ VALUE (IL:PROMPTFORWORD NIL NIL NIL AMUSED.MAIN.WINDOW))
                       (TERPRI AMUSED.MAIN.WINDOW)
                       (RETURN VALUE))))))

(AM.GET.INPUT.NUMBER
  (LAMBDA NIL                                         (IL:* IL:\; "Edited 8-Mar-89 07:58 by MAFoltman")
          (BLOCK AM.GET.INPUT.NUMBER
           (IL:* IL:|;;| " Retrieve the input entered into the main AMUSED window.")
           (IL:* IL:|;;| " A symbol is returned from the read.")
           (IL:* IL:|;;| "")

                 (PROG (VALUE)
                       (SETQ VALUE (READ-FROM-STRING (IL:PROMPTFORWORD NIL NIL NIL AMUSED.MAIN.WINDOW)
                                                     ))
                       (TERPRI AMUSED.MAIN.WINOOW)
                       (RETURN VALUE))))))

(PRINT.QUESTION
  (LAMBDA (CLAUSE.LIST)
          (BLOCK PRINT.QUESTION (PROG (QUESTION.LIST)
                                      (SETQ QUESTION.LIST (CADR (ASSOC (CADAR CLAUSE.LIST)
```

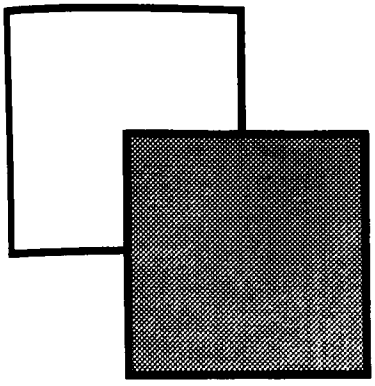
```
                                                                    GLOBAL.QUESTION.LIST)))
                                        (PRINC.STRING "'")
                                        (PRINC.STRING (CADR (ASSOC (CADR CLAUSE.LIST)
                                                                   QUESTION.LIST)))
                                        (PRINC.STRING "'")))))

(AM.ERROR
  (LAMBDA (STRING)
          (BLOCK AM.ERROR (PRINT.STRING STRING))))
)


(IL:* IL:|;;| "")


(IL:* IL:|;;| "String manipulations")

(IL:DEFINEQ

(REVERSE.STRING
  (LAMBDA (STRING)                                  (IL:* IL:\; "Edited 27-Feb-89 16:35 by MAFoltman")
          (BLOCK REVERSE.STRING
           (IL:* IL:|;;| " Reverse.string takes a given string as input and reverses the string ")
           (IL:* IL:|;;| " front to back, and returns the string in reversed order.")
           (IL:* IL:|;;| "")

                (PROG (NEW.STRING X)
                      (SETQ NEW.STRING "")
                      (DOTIMES (X (IL:NCHARS STRING))
                             (SETQ NEW.STRING (IL:CONCAT NEW.STRING (IL:GLC STRING))))
                      (RETURN NEW.STRING)))))

(AM.ISOLATE.KEY.TEXT
  (LAMBDA (STRING FRONT.CHAR.NUM BACK.CHAR.NUM)
                                                    (IL:* IL:\; "Edited 27-Feb-89 16:37 by MAFoltman")
          (BLOCK AM.ISOLATE.KEY.TEXT
           (IL:* IL:|;;| "The string given is truncated by the given number of ")
           (IL:* IL:|;;| "characters in the front and  the specified number of")
           (IL:* IL:|;;| " characters from the back. The resulting truncated string")
           (IL:* IL:|;;| " is returned.")
           (IL:* IL:|;;| "")

                (PROG (DUMMYSTRING)
                      (SETQ DUMMYSTRING STRING)
                      (RETURN (REVERSE.STRING (IL:SUBSTRING (REVERSE.STRING (IL:SUBSTRING
                                                                              DUMMYSTRING
                                                                              FRONT.CHAR.NUM
                                                                              NIL))
                                                            BACK.CHAR.NUM NIL)))))))
)


(IL:* IL:|;;| "")


(IL:* IL:|;;| "Functions for Global variable checking inside of rules")

(IL:DEFINEQ

(SAME.GLOBAL
  (LAMBDA (PARAMETER VALUE)                         (IL:* IL:\; "Edited 6-Mar-89 17:51 by MAFoltman")
          (BLOCK SAME.GLOBAL
           (IL:* IL:|;;| "Check the value of a global parameter or question with a given value. ")
           (IL:* IL:|;;| " This global check takes the place of the local TMYCIN call SAME or")
           (IL:* IL:|;;| " NOTSAME.")
           (IL:* IL:|;;| "")

                (PROG (RESULT)
```

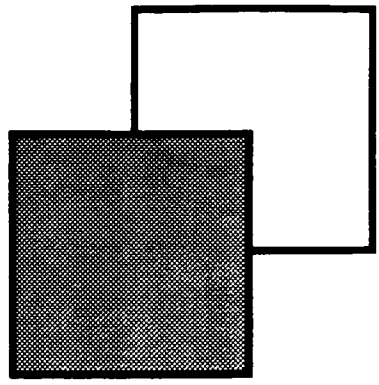
```
                    (CONO
                       ((ANO (LISTP PARAMETER)
                             (EQUAL (LENGTH PARAMETER)
                                    2))
                        (CONO
                           ((EQUALP (CAR PARAMETER)
                                    VALUE)
                            (SETQ RESULT (CAOR PARAMETER)))
                           (T (SETQ RESULT -1.0))))
                       (T (CONO
                             ((EQUALP PARAMETER VALUE)
                              (SETQ RESULT 1.0))
                             (T (SETQ RESULT -1.0)))))
                    (CONO
                       ((NOT (NULL PARAMETER))
                        (AOO.TO.HISTORY.LIST RESULT)))
                    (RETURN RESULT)))))
```

```
(SPECIFIC.QUESTION
  (LAMBOA (KEY QUESTION.NUMBER)                    (IL:* IL:\; "Edited 8-Mar-89 08:06 by MAFoltman")
          (BLOCK SPECIFIC.QUESTION
           (IL:* IL:|;;| "   Specific.question first checks to see if the question being")
           (IL:* IL:|;;| " requested has been asked before.  If so, then the resulting ")
           (IL:* IL:|;;| " answer and certainty factor are returned.  Otherwise, it ")
           (IL:* IL:|;;| " finds the category of the question and loads the specific ")
           (IL:* IL:|;;| " question from the currently loaded global question data-")
           (IL:* IL:|;;| " base.")
           (IL:* IL:|;;| "    It presents the question to the user, prompting for an ")
           (IL:* IL:|;;| " answer, and returns the answer and associated certainty ")
           (IL:* IL:|;;| " factor to the calling function. The certainty factor is calculated ")
           (IL:* IL:|;;| " from the user entering his/her confidence level on the ")
           (IL:* IL:|;;| " question's answer.")
           (IL:* IL:|;;| "   The KEY value specified determines the category in the ")
           (IL:* IL:|;;| " database where the question comes from.")
           (IL:* IL:|;;| "   The QUESTION.NUMBER specifies the actual question in ")
           (IL:* IL:|;;| " the KEY category.")
           (IL:* IL:|;;| "")

                 (PROG (QUESTION.LIST QUESTION ANSWER TOPIC TOPIC.LIST ASKEO.BEFORE TOPIC)
           (IL:* IL:|;;| "")

                       (SETQ ASKEO.BEFORE (OOLIST (TOPIC (OOLIST (TOPIC QUESTIONS.ASKEO TOPIC.LIST)
                                                                 (CONO
                                                                    ((EQUAL (CAR TOPIC)
                                                                            KEY)
                                                                     (SETF TOPIC.LIST (CONS TOPIC
                                                                                            TOPIC.LIST)
                                                                     ))))
                                                          ASKEO.BEFORE)
                                                 (CONO
                                                    ((EQUAL (CAOR TOPIC)
                                                            QUESTION.NUMBER)
                                                     (SETQ ASKEO.BEFORE TOPIC)))))
                       (CONO
                          ((NOT (NULL ASKEO.BEFORE))
                           (AOO.TO.HISTORY.LIST (LIST KEY QUESTION.NUMBER ANSWER))
                           (RETURN (COOR ASKEO.BEFORE)))
                          (T (SETQ QUESTION.LIST (CAOR (ASSOC KEY GLOBAL.QUESTION.LIST)))
                             (SETQ QUESTION (CAOR (ASSOC QUESTION.NUMBER QUESTION.LIST)))
                             (SETQ ANSWER (CREATE.MENU QUESTION '(YES NO) T))
                             (SETQ CF.VALUE (COLLECT.USER.CERTAINTY))
                             (AOO.TO.HISTORY.LIST (LIST KEY QUESTION.NUMBER ANSWER))
                             (SETQ QUESTIONS.ASKEO (CONS (LIST KEY QUESTION.NUMBER ANSWER CF.VALUE)
                                                         QUESTIONS.ASKEO))
                             (RETURN (LIST ANSWER CF.VALUE)))))))))
)

(IL:* IL:|;;| "")
```

```
(IL:* IL:|;;| " Functions for seting the global variables from the state of the freemenu options.")

(IL:DEFINEQ

(AM.SET.GLOBAL.OPTIONS
  (LAMBDA (ITEM WINDOW BUTTONS)                    (IL:* IL:\; "Edited 27-Feb-89 16:46 by MAFoltman")
          (BLOCK AM.SET.GLOBAL.OPTIONS
           (IL:* IL:|;;| "This function is called from the Diagnose options freemenu. Once the ")
           (IL:* IL:|;;| " options have been set and Complete! has been selected. The options")
           (IL:* IL:|;;| " of the freemenu are then read and the corresponding global variables")
           (IL:* IL:|;;|
         " are set with the latest values. Several other operations are also executed")
           (IL:* IL:|;;| " dependent upon the options selected.")
           (IL:* IL:|;;|
         " If the ENABLE.LOG option is set, then the Log.file is created and opened.")
           (IL:* IL:|;;| "")

                 (PROG (OPTION.SELECTIONS)
                       (SETQ OPTION.SELECTIONS (IL:FM.GETSTATE DIAGNOSE.OPTIONS))
                       (SETQ LINKER.USED (COND
                                           ((IL:LISTGET OPTION.SELECTIONS 'LINK)
                                            'LINK)
                                           ((IL:LISTGET OPTION.SELECTIONS 'LINK8051)
                                            'LINK8051)))
                       (COND
                          ((EQUAL LINKER.USED 'LINK)
                           (DEFINE.LINK.QUESTIONS)
                           (SETQ GLOBAL.QUESTION.LIST LINK.QUESTIONS))
                          ((EQUAL LINKER.USED 'LINK8051)
                           (DEFINE.LINK8051.QUESTIONS)
                           (SETQ GLOBAL.QUESTION.LIST LINK8051.QUESTIONS)))
                       (SETQ PROCESSOR.USED (COND
                                              ((IL:LISTGET OPTION.SELECTIONS '8085)
                                               '8085)
                                              ((IL:LISTGET OPTION.SELECTIONS '8051)
                                               '8051)))
                       (COND
                          ((OR (AND (EQUAL LINKER.USED 'LINK8051)
                                    (EQUAL PROCESSOR.USED '8085))
                               (AND (EQUAL LINKER.USED 'LINK)
                                    (EQUAL PROCESSOR.USED '8051)))
                           (AM.ERROR
                                 "Please select again. The tool and processor are not compatible")
                           (IL:REDISPLAYW DIAGNOSE.OPTIONS)
                           (SETQ CONTINUE? NIL)
                           (AM.SETUP.DIAGNOSE))
                          (T (SETQ CONTINUE? T)))
                       (COND
                          (CONTINUE? (SETQ ENABLE.LOG (IL:LISTGET OPTION.SELECTIONS 'ENABLELOG))
                                 (COND
                                    (ENABLE.LOG (CREATE.LOG.FILE)))
                                 (AM.DIAGNOSE.MODE)))))))
)


(IL:* IL:|;;| "")


(IL:* IL:|;;| " User Interface functions")

(IL:DEFINEQ

(CREATE.LOG.FILE
  (IL:LAMBDA NIL                                   (IL:* IL:\; "Edited 3-Apr-89 14:34 by MAFoltman")
    (PROG (FILE AMUSED.DATE)
          (IL:* IL:|;;| " Create a log file which will capture the interaction")
          (IL:* IL:|;;| " of the diagnose session. This file is a variable to which everything")
          (IL:* IL:|;;| " is appended. The file is written out at the end of a session and")
          (IL:* IL:|;;| " can be printed by the user.")
          (IL:* IL:|;;| "")
```

```
          (SETQ LDG.FILE NIL))))

(CREATE.MENU
  (LAMBDA (TITLE MENU.ITEMS REPLICATE?)          (IL:* IL:\; "Edited 27-Feb-89 16:50 by MAFoltman")
          (BLOCK CREATE.MENU
           (IL:* IL:|;;| "Create a menu with the given title and menu items.")
           (IL:* IL:|;;| " Return the answer received from the question.")
           (IL:* IL:|;;| " If the Replicate? flag is set, then the values are printed")
           (IL:* IL:|;;| " to the main  AMUSED window.")
           (IL:* IL:|;;| "")

                  (PROG (ANSWER)
                        (COND
                           (REPLICATE? (PRINT.STRING TITLE)))
                        (IL:|repeatuntil| (SETQ ANSWER
                                                (IL:MENU (IL:|create| IL:MENU
                                                                      IL:ITEMS IL:← MENU.ITEMS
                                                                      IL:TITLE IL:← TITLE
                                                                      IL:CENTERFLG IL:← T))))
                        (COND
                           (REPLICATE? (PRINT.STRING ANSWER)))
                        (RETURN ANSWER)))))

(COLLECT.USER.CERTAINTY
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 3-Apr-89 14:35 by MAFoltman")
          (BLOCK COLLECT.USER.CERTAINTY
                 (PROG (CF.VALUE CERTAINTY.TERMS)
                       (IL:|repeatuntil| (SETQ CERTAINTY.TERMS
                                               (CREATE.MENU
                                                      "     How certain of this answer are you?"
                                                      '("Not certain" "Somewhat Certain"
                                                             "Fairly Certain" "Very Sure"
                                                             "Positive") T)))
                       (COND
                          ((EQUAL CERTAINTY.TERMS "Not certain")
                           (SETQ CF.VALUE 0.1))
                          ((EQUAL CERTAINTY.TERMS "Somewhat Certain")
                           (SETQ CF.VALUE 0.5))
                          ((EQUAL CERTAINTY.TERMS "Fairly Certain")
                           (SETQ CF.VALUE 0.7))
                          ((EQUAL CERTAINTY.TERMS "Very Sure")
                           (SETQ CF.VALUE 0.9))
                          ((EQUAL CERTAINTY.TERMS "Positive")
                           (SETQ CF.VALUE 1.0)))
                       (RETURN CF.VALUE)))))
)
(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADOTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA COLLECT.USER.CERTAINTY CREATE.MENU AM.SET.GLOBAL.OPTIONS SPECIFIC.QUESTION
                    SAME.GLOBAL AM.ISOLATE.KEY.TEXT REVERSE.STRING AM.ERROR PRINT.QUESTION
                    AM.GET.INPUT.NUMBER AM.GET.INPUT AM.NEW.LINE PRINC.STRING PRINT.STRING)

)
(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL (2423 6558 (PRINT.STRING 2436 . 3391) (PRINC.STRING 3393 . 4156) (AM.NEW.LINE 4158
4739) (AM.GET.INPUT 4741 . 5266) (AM.GET.INPUT.NUMBER 5268 . 5890) (PRINT.QUESTION 5892 . 6468) (
AM.ERROR 6470 . 6556)) (6631 8404 (REVERSE.STRING 6644 . 7291) (AM.ISOLATE.KEY.TEXT 7293 . 8402)) (
8511 13210 (SAME.GLOBAL 8524 . 9778) (SPECIFIC.QUESTION 9780 . 13208)) (13345 16247 (
AM.SET.GLOBAL.OPTIONS 13358 . 16245)) (16325 19423 (CREATE.LOG.FILE 16338 . 16902) (CREATE.MENU 16904
 . 18064) (COLLECT.USER.CERTAINTY 18066 . 19421)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>AMUSED-WINDOWS.;48

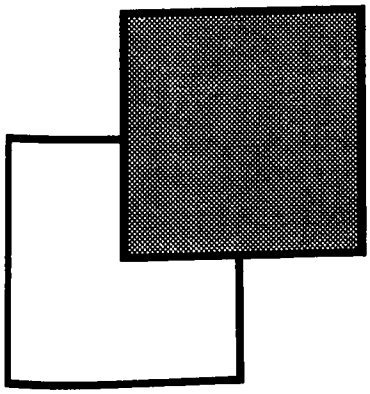
Created 1-May-89 8:45:20

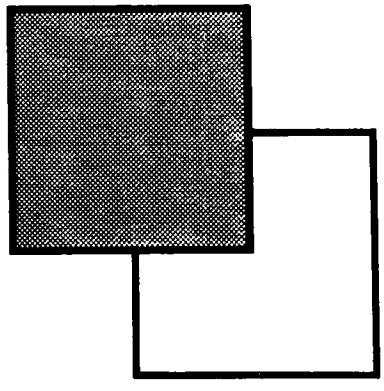
Printed 1-May-89 8:46:46

12 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(DEFINE-FILE-INFO ▮PACKAGE "XCL-USER" ▮READTABLE "XCL" ▮BASE 10)
(IL:FILECREATED " 3-Apr-89 16:42:52" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-WINDOWS.\;48 50152

      IL:|changes| IL:|to:|  (IL:FNS AMUSED.ICONFN AM.CLOSE.FILES)

      IL:|previous| IL:|date:| "20-Mar-89 13:24:48" IL:{DSK}<LISPFILES>THESIS>CODE>AMUSED-WINDOWS.\;46
)


(IL:PRETTYCOMPRINT IL:AMUSED-WINDOWSCOMS)

(IL:RPAQQ IL:AMUSED-WINDOWSCOMS
          ((IL:P (IN-PACKAGE 'IL:XCL-USER))
           (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                  IL:AMUSED-WINDOWS)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| " Global Variables used for Amused")

           (IL:VARS (LINKER.USED NIL)
                  (ERROR.TYPE NIL)
                  (ENABLE.LOG NIL)
                  (GLOBAL.QUESTION.LIST NIL)
                  (FOUND? NIL)
                  (QUESTIONS.ASKED NIL)
                  (PROCESSOR.USED NIL))

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| " Diagnose mode")

           (IL:FNS AM.SETUP.DIAGNOSE AM.DIAGNOSE.MODE AM.REPEAT.DIAGNOSE)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| " Assistance Mode")

           (IL:FNS AM.SETUP.ASSIST)

          (IL:* IL:|;;| "")


          (IL:* IL:|;;| " Assistance Functions with User information")

           (IL:FNS AM.ASSIST.INVALID.SWITCHES AM.ASSIST.VALID.SWITCHES AM.ASSIST.USERCM.OPTIONS
                  AM.ASSIST.USERCM.SETUP AM.ASSIST.SEARCH.PATH.ACTIVE AM.ASSIST.SEARCH.PATH.RESET
                  AM.ASSIST.LINK.FORMAT AM.ASSIST.LINK.BASICS AM.ASSIST.PSIZE.FORMAT
                  AM.ASSIST.PSIZE.DEFN AM.ASSIST.PSIZE.SWITCHES)

          (IL:* IL:|;;| " ")


          (IL:* IL:|;;| "Functions to bring up basic window")

           (IL:FNS AMUSED.START AMUSED.OPEN.BASIC.CMD.WINDOW AM.WINDOW.SCROLL RESHAPE.AMUSED.WINDOW
                  REPAINT.AMUSED.WINDOW AM.CLOSE.FILES AMUSED.ICONFN)

          (IL:* IL:|;;| "")

           (IL:BITMAPS AMUSEDBITS AMUSEDMASK)

          (IL:* IL:|;;| "(ADDVARS (BACKGROUNDMENUCOMMANDS (\"AMUSED\" '(XCL::AMUSED.START) \"Opens the main window for A
MUSED diagnostics\")))")


          (IL:* IL:|;;| "")

           (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS
                  (IL:ADDVARS (IL:NLAMA)
                         (IL:NLAML)
```

```
                        (IL:LAMA AMUSED.ICONFN AM.CLOSE.FILES REPAINT.AMUSED.WINDOW
                               RESHAPE.AMUSED.WINDOW AM.WINDOW.SCROLL AMUSED.OPEN.BASIC.CMD.WINDOW
                               AMUSED.START AM.ASSIST.PSIZE.SWITCHES AM.ASSIST.PSIZE.DEFN
                               AM.ASSIST.PSIZE.FORMAT AM.ASSIST.LINK.BASICS AM.ASSIST.LINK.FORMAT
                               AM.ASSIST.SEARCH.PATH.RESET AM.ASSIST.SEARCH.PATH.ACTIVE
                               AM.ASSIST.USERCM.SETUP AM.ASSIST.USERCM.OPTIONS
                               AM.ASSIST.VALID.SWITCHES AM.ASSIST.INVALID.SWITCHES AM.SETUP.ASSIST
                               AM.REPEAT.DIAGNOSE AM.DIAGNOSE.MODE AM.SETUP.DIAGNOSE)))))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:AMUSED-WINDOWS IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL"
                                                                :BASE 10))


(IL:* IL:|;;| "")


(IL:* IL:|;;| " Global Variables used for Amused")

(IL:RPAQQ LINKER.USED NIL)

(IL:RPAQQ ERROR.TYPE NIL)

(IL:RPAQQ ENABLE.LOG NIL)

(IL:RPAQQ GLOBAL.QUESTION.LIST NIL)

(IL:RPAQQ FOUND? NIL)

(IL:RPAQQ QUESTIONS.ASKED NIL)

(IL:RPAQQ PROCESSOR.USED NIL)


(IL:* IL:|;;| "")


(IL:* IL:|;;| " Diagnose mode")

(IL:DEFINEQ

(AM.SETUP.DIAGNOSE
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 9-Mar-89 14:48 by MAFoltman")
        (BLOCK AM.SETUP.DIAGNOSE
         (IL:* IL:|;;| "Open the diagnose option freemenu, and prompt")
         (IL:* IL:|;;| " the user on the next steps.")
         (IL:* IL:|;;| "")

              (IL:ATTACHWINDOW DIAGNOSE.OPTIONS AMUSED.MAIN.WINDOW 'IL:TOP 'IL:JUSTIFY '
                    IL:LOCALCLOSE)
              (IL:CLEARW AMUSED.MAIN.WINDOW)
              (IL:REDISPLAYW DIAGNOSE.OPTIONS)
              (PRINT.STRING "Please select the correct options for this session.")
              (PRINT.STRING "Select Complete! when finished."))))

(AM.DIAGNOSE.MODE
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 8-Mar-89 08:13 by MAFoltman")
        (BLOCK AM.DIAGNOSE.MODE
         (IL:* IL:|;;| " Once the global options are set, determine what type of")
         (IL:* IL:|;;| " error is being addressed. The correct rule base is loaded")
         (IL:* IL:|;;| " and the diagnosing session is started.")
         (IL:* IL:|;;| "")

              (PROG (OPTIONS.SET)
                    (SETQ ERROR.TYPE (CREATE.MENU "What type of error is it?"
                                                  '(WARNING ERROR FATAL) T))
                    (SETF (GET 'SOLUTION 'RULES)
```

```
                    NIL)
                (CONO
                   ((EQUAL LINKER.USED 'LINK)
                    (CASE ERROR.TYPE (WARNING (PRINT.STRING "Loading Rule base for warnings..."
                                                            )
                                              (DEFINE.RULE.BASE.WARNING))
                          (ERROR (PRINT.STRING "Loading Rule Base for errors...")
                                 (DEFINE.RULE.BASE.ERROR))
                          (FATAL (PRINT.STRING "Loading Rule Base for fatal errors...")
                                 (DEFINE.RULE.BASE.FATAL))))
                   (T (CASE ERROR.TYPE (WARNING (PRINT.STRING
                                                  "Loading Rule base for B051 warnings..."
                                                  )
                                                (DEFINE.RULE.BASE.B051.WARNING))
                            (ERROR (PRINT.STRING "Loading Rule Base for B051 errors...")
                                   (DEFINE.RULE.BASE.B051.ERROR))
                            (FATAL (PRINT.STRING "Loading Rule Base for fatal B051 errors...")
                                   (DEFINE.RULE.BASE.B051.FATAL)))))
                (SETF (GET 'SOLUTION 'RULES)
                      ALLRULES)
                (SETQ FOUND? NIL)
                (SETQ QUESTIONS.ASKED NIL)
                (DOCONSULT 'ERROR)
                (AM.REPEAT.DIAGNOSE)))))
```

```
(AM.REPEAT.DIAGNOSE
  (LAMBDA NIL                                   (IL:* IL:\; "Edited 8-Mar-89 08:14 by MAFoltman")
        (BLOCK AM.REPEAT.DIAGNOSE (PROG NIL
                                        (SETQ REPEAT? (CREATE.MENU
                                                        "Do you have another message to diagnose?"
                                                          '(YES NO) T))
                                        (CONO
                                           ((EQUAL REPEAT? 'YES)
                                            (AM.SETUP.DIAGNOSE))
                                           (T (AM.CLOSE.FILES)))))))
)

(IL:* IL:|;;| "")


(IL:* IL:|;;| " Assistance Mode")

(IL:DEFINEQ

(AM.SETUP.ASSIST
                                              (IL:* IL:\; "Edited 6-Mar-89 15:49 by MAFoltman")
  (LAMBDA NIL
        (BLOCK AM.SETUP.ASSIST (PROG NIL
                                     (SETQ ASSIST.TYPE (CREATE.MENU
                                                         "What category would like assistance for?"
                                                           '("Switches" "User.CM" "Search Path"
                                                                  "Link Format" "PSIZE") T))

                                     (CONO
                                        ((EQUAL ASSIST.TYPE '"Switches")
                                         (SETQ ITEMS '("Invalid Switch Combinations"
                                                       "Valid Switches")))
                                        ((EQUAL ASSIST.TYPE '"User.CM")
                                         (SETQ ITEMS '("User.CM Options" "Setting up the User.CM")))
                                        ((EQUAL ASSIST.TYPE '"Search Path")
                                         (SETQ ITEMS '("Active Search Path"
                                                       "Resetting the Search Path")))
                                        ((EQUAL ASSIST.TYPE '"Link Format")
                                         (SETQ ITEMS '("General Link Format" "Basic Link Needs")))
                                        ((EQUAL ASSIST.TYPE '"PSIZE")
                                         (SETQ ITEMS '("PSIZE Format" "PSIZE Definition"
                                                       "PSIZE Switches")))
                                        (T (SETQ ITEMS '("None"))))
                                     (SETQ SPECIFIC.ASSIST.TYPE (CREATE.MENU
                                                "Which aspect in particular are you interested in?"
                                                                  ITEMS T))

                                     (CONO
```

```
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Invalid Switch Combinations")
                                         (AM.ASSIST.INVALID.SWITCHES))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Valid Switches")
                                         (AM.ASSIST.VALID.SWITCHES))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "User.CM Options")
                                         (AM.ASSIST.USERCM.OPTIONS))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Setting up the User.CM")
                                         (AM.ASSIST.USERCM.SETUP))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Active Search Path")
                                         (AM.ASSIST.SEARCH.PATH.ACTIVE))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Resetting the Search Path")
                                         (AM.ASSIST.SEARCH.PATH.RESET))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "General Link Format")
                                         (AM.ASSIST.LINK.FORMAT))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "Basic Link Needs")
                                         (AM.ASSIST.LINK.BASICS))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "PSIZE Format")
                                         (AM.ASSIST.PSIZE.FORMAT))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "PSIZE Definition")
                                         (AM.ASSIST.PSIZE.DEFN))
                                        ((EQUAL SPECIFIC.ASSIST.TYPE "PSIZE Switches")
                                         (AM.ASSIST.PSIZE.SWITCHES)))))))
)

(IL:* IL:|;;| "")

(IL:* IL:|;;| " Assistance Functions with User information")

(IL:DEFINEQ

(AM.ASSIST.INVALID.SWITCHES
  (LAM8DA NIL                                   (IL:* IL:\; "Edited 8-Mar-89 08:15 by MAFoltman")
        (BLOCK AM.ASSIST.INVALID.SWITCHES (PRINT.STRING
                        "  The following switch combinations are invalid for Link8051 and Link:"
                                          )
                (PRINT.STRING "     > The l switch off with any of the d,m,n, s, and/or v")
                (PRINT.STRING "       switches being turned on.")
                (PRINT.STRING
                        "     > The h and i switches are exclusive. An error results if both of these"
                        )
                (PRINT.STRING "       switches are on. If both are off, the image file")
                (PRINT.STRING "       format defaults to Sigma mif.")
                (PRINT.STRING
                        "     > the n and v switches are not exclusive of each other.  If both are")
                (PRINT.STRING
                        "       turned on, two symbol table listings will be written, one sorted by")
                (PRINT.STRING "       name, and the other sorted by value.")
                (PRINT.STRING "  For the Link tool, the y switch causes warning about the use of")
                (PRINT.STRING "       different creators to be issued.  If -y is specified, these ")
                (PRINT.STRING "       warnings are disabled."))))

(AM.ASSIST.VALID.SWITCHES
  (LAM8DA NIL                                   (IL:* IL:\; "Edited 6-Mar-89 18:30 by MAFoltman")
        (BLOCK AM.ASSIST.VALID.SWITCHES (PROG (TOOL)
                                              (SETQ TOOL (CREATE.MENU "Switches for which Tool?"
                                                                      '("Link" "Link8051") T))
                                              (COND
                                                 ((EQUAL TOOL "Link")
                                                  (PRINT.STRING
                                     "  The valid switches for Link are a, b, c, d, e, g, h, i, k, l,"
                                                                )
                                                  (PRINT.STRING
                                     "     m, n, p, s, v, w, x, y, and z. The meanings of these switches"
                                                                )
                                                  (PRINT.STRING
                                     "     are listed below. A switch is turned off by preceding it with"
                                                                )
                                                  (PRINT.STRING
                                     "     a   or a  .  Default values for switches may be specified in "
```

```
                                )
                            (PRINT.STRING
"     the user.cm. ( See assistance under User.cm) The default settings"
                                )
                            (PRINT.STRING
                 "     of the switches are equivalent to a switch list of "
                                )
                            (PRINT.STRING
                                    "     /-a-b-c-de-g-h-i-klmn-ps-vw-xy-z."))
                           (T (PRINT.STRING
   "  The valid switches for Link8051 are a, b, c, d, e, g, h, i, k, l,"
                                )
                             (PRINT.STRING
        "     m, n, p, s, v, w, and x. The meanings of these switches"
                                )
                             (PRINT.STRING
    "     are listed below. A switch is turned off by preceding it with"
                                )
                             (PRINT.STRING
    "     a   or a   .  Oefault values for switches may be specified in "
                                )
                             (PRINT.STRING
"     the user.cm. ( See assistance under User.cm) The default settings"
                                )
                             (PRINT.STRING
                 "     of the switches are equivalent to a switch list of "
                                )
                             (PRINT.STRING
                                    "     /-a-b-c-de-g-h-i-klmn-ps-vw-x.")))
                         (PRINT.STRING " Switch      Meaning (default)")
                         (CONO
                            ((EQUAL TOOL "Link")
                             (PRINT.STRING
                                    "   a       Produce abs info file (-a)."))
                            (T (PRINT.STRING
                              "   a       Produce absolute listing files (-a)."
                                    )))
                         (PRINT.STRING
                  "   b       Use EBCOIC rather than ASCII character set (-b)."
                                )
                         (PRINT.STRING
                "   c       Create list of global symbols in the image files (-c)."
                                )
                         (PRINT.STRING
                "   d       Write dependency listing to the link listing file (-d)."
                                )
                         (PRINT.STRING
                  "   e       Write diagnostics to the error log file (e)."
                                )
                         (PRINT.STRING
                                "   g       Produce global symbols file (-g).")
                         (PRINT.STRING
                     "   h       Write image files in Intel hex format, rather "
                                )
                         (PRINT.STRING "           than Sigma mif (-h).")
                         (PRINT.STRING
                   "   i       Write image files in Intel object format, rather "
                                )
                         (PRINT.STRING "           than Sigma mif (-i).")
                         (CONO
                            ((EQUAL TOOL "Link")
                             (PRINT.STRING
                   "   k       Keep .abs, .hex, .intelObj, .mif, and .sym files "
                                )
                             (PRINT.STRING "           if link aborts (-k). "))
                            (T (PRINT.STRING
                      "   k       Keep .hex, .intelObj, .mif, and .sym files "
                                )
                               (PRINT.STRING "           if link aborts (-k). "
                                )))
                         (PRINT.STRING
                                "   l       Produce link listing file (l).")
                         (PRINT.STRING
                     "   m       Write memory map to the link listing file (m)."
```

```
                                        )
                                   (PRINT.STRING
                              "   n       Write symbol table listing to the link listing file,"
                                        )
                                   (PRINT.STRING "           sorted by name (n).")
                                   (PRINT.STRING
                                "   p       Terminate processing if errors were found in "
                                        )
                                   (PRINT.STRING "           previous command (-p).")
                                   (PRINT.STRING
                           "   s       Report input files with non-zero error severity levels (s)."
                                        )
                                   (PRINT.STRING
                            "   v       Write symbol table listing to the link listing file, (-b)."
                                        )
                                   (PRINT.STRING "           sorted by value (-v).")
                                   (PRINT.STRING
                                           "   w       Report warning messages (w).")
                                   (PRINT.STRING
                                      "   x       Produce cross reference listing file (-x)."
                                        )
                                   (COND
                                      ((EQUAL TOOL "Link")
                                       (PRINT.STRING
                               "   y       Issue warnings about the use of different creators (y)."
                                             )
                                       (PRINT.STRING
                            "   z       Produce psize listing file (See Assistance on PSIZE) (-z)."
                                             )))))))
```

(AM.ASSIST.USERCM.OPTIONS

```
(LAMBDA NIL                                   (IL:* IL:\; "Edited 9-Mar-89 12:26 by MAFoltman")
       (BLOCK AM.ASSIST.USERCM.OPTIONS (PRINT.STRING
                                        "      The user.cm slice for Link may contain values for"
                                        )
              (PRINT.STRING " Checksum, Hex, IntelObj, Map, Mif, Psize, RecordSize,")
              (PRINT.STRING " StreamWindowSize, or Switches.")
              (AM.NEW.LINE)
              (PRINT.STRING "      The options available for Link8051 include Checksum,")
              (PRINT.STRING " Map, RecordSize, or Switches.")
              (AM.NEW.LINE)
              (PRINT.STRING "      Section 2.4 of both the Link and Link8051 User's Manual")
              (PRINT.STRING " contain more details on the user.cm options and values.")))
```

(AM.ASSIST.USERCM.SETUP

```
(LAMBDA NIL                                   (IL:* IL:\; "Edited 9-Mar-89 12:22 by MAFoltman")
       (BLOCK AM.ASSIST.USERCM.SETUP (COND
                                     ((EQ LINKER.USED 'LINK)
                                      (PRINT.STRING
                                          "      User defaults for certain input values may be "
                                          )
                                      (PRINT.STRING
                                         " specified in the [Link] section of the user.cm.  The"
                                          )
                                      (PRINT.STRING
                                      " map file, checksum file, psize file, image input file, "
                                          )
                                      (PRINT.STRING
                                         " and switches may be specified here.  Entries in the "
                                          )
                                      (PRINT.STRING " [Link] slice of the user.cm take the form"
                                          )
                                      (PRINT.STRING "       key:value")
                                      (PRINT.STRING
                                       " where key is one of Checksum, Hex, IntelObj, Map, Mif,"
                                          )
                                      (PRINT.STRING
                                     " Psize, RecordSize, StreamWindowSize, or Switches. Section"
                                          )
                                      (PRINT.STRING
                                     " 2.4 of the Link User's manual shows the necessary values "
                                          )
                                      (PRINT.STRING
                                     " for each of these fields.  An example user.cm slice might"
```

```
                                    )
                                (PRINT.STRING " look like the following:")
                                (PRINT.STRING "      [Link]")
                                (PRINT.STRING "      Map: PrintControl.map")
                                (PRINT.STRING "      Switches: ag-ebv")
                                (PRINT.STRING "      Mif: Base")
                                (PRINT.STRING
                             "      Please talk with other members in your group to ensure"
                                    )
                                (PRINT.STRING
                            " that the user.cm slice is set up to the project's standards."
                                    ))
                               (T (PRINT.STRING
                                    "      User defaults for certain input values may be "
                                    )
                                  (PRINT.STRING
                                " specified in the [Link8051] section of the user.cm.  The"
                                    )
                                  (PRINT.STRING
                                 " map file, checksum file, and switches may be specified "
                                    )
                                  (PRINT.STRING
                                  " here.  Entries in the [Link8051] slice of the user.cm "
                                    )
                                  (PRINT.STRING " take the form ")
                                  (PRINT.STRING "       key:value")
                                  (PRINT.STRING
                             " where key is one of Checksum, Map, RecordSize, or Switches."
                                    )
                                  (PRINT.STRING
                                   " Section 2.4 of the Link User's manual shows the necessary"
                                    )
                                  (PRINT.STRING
                                  " values for each of these fields.  An example user.cm slice"
                                    )
                                  (PRINT.STRING " might look like the following:")
                                  (PRINT.STRING "      [Link8051]")
                                  (PRINT.STRING "      Map: PrintControl.map")
                                  (PRINT.STRING "      Switches: ag-ebv")
                                  (AM.NEW.LINE)
                                  (PRINT.STRING
                             "      Please talk with other members in your group to ensure"
                                    )
                                  (PRINT.STRING
                            " that the user.cm slice is set up to the project's standards."
                                    ))))))
```

(AM.ASSIST.SEARCH.PATH.ACTIVE

```
(LAMBDA NIL (BLOCK AM.ASSIST.SEARCH.PATH.ACTIVE (PRINT.STRING
                                    "      The search path tool in XDE (XEROX Development"
                                      )
                    (PRINT.STRING " Environment) is capable  of creating and destroying")
                    (PRINT.STRING " directories of the  workstation.  These directories ")
                    (PRINT.STRING " can be arranged into an active list of directories.")
                    (AM.NEW.LINE)
                    (PRINT.STRING "      To see the active set of directories, the search")
                    (PRINT.STRING " path, on a  workstation,  open  the  search path tool.")
                    (PRINT.STRING " The top category in the window  lists the directories ")
                    (PRINT.STRING " which are currently on the active list.")
                    (AM.NEW.LINE))))
```

(AM.ASSIST.SEARCH.PATH.RESET

```
(LAMBDA NIL (BLOCK AM.ASSIST.SEARCH.PATH.RESET (PRINT.STRING
                                     "      To reset the active list of directories on "
                                      )
                    (PRINT.STRING " the search path, open the Search Path tool. Select")
                    (PRINT.STRING " Pop! to remove the top element of the active list.")
                    (AM.NEW.LINE)
                    (PRINT.STRING "      To add a directory to the top of the search")
                    (PRINT.STRING " path, the directory name must be supplied in the ")
                    (PRINT.STRING " Oirectory: slot and Push! is selected.  The speci-")
                    (PRINT.STRING " fied directory will then be placed on the top of the")
                    (PRINT.STRING " active search path.")
                    (AM.NEW.LINE)
```

```
                    (PRINT.STRING "      More information may be obtained fromthe XOE ")
                    (PRINT.STRING " User's guide.")
                    (AM.NEW.LINE))))
```

(AM.ASSIST.LINK.FORMAT

```
  (LAMBOA NIL (BLOCK AM.ASSIST.LINK.FORMAT (PRINT.STRING
                                    "      The Link or Link8051 tool is executed in the executive"
                                        )
                    (PRINT.STRING " window with the basic command statement,")
                    (PRINT.STRING "      LINK filename/switches")
                    (PRINT.STRING " for Link, and ")
                    (PRINT.STRING "      LINK8051 filename/switches")
                    (PRINT.STRING " for Link805, where Filename is the name of the link ")
                    (PRINT.STRING " file and the switches are a string of characters, with each")
                    (PRINT.STRING " character being interpreted as a separate switch.")
                    (AM.NEW.LINE)
                    (PRINT.STRING "      The basic link file has a number of different")
                    (PRINT.STRING " sections, each with different meaning.  The ")
                    (PRINT.STRING " first and main section of the link file begins with")
                    (PRINT.STRING "      LINK   ")
                    (PRINT.STRING " and ends with ")
                    (PRINT.STRING "      ENO LINK")
                    (PRINT.STRING " Two portions exist within this statement, and they ")
                    (PRINT.STRING " are the SPECS portion and the ROOT/OVERLAY portion. These")
                    (PRINT.STRING " two sections have the same statement format as")
                    (PRINT.STRING " the Link statement.  The SPECS section is used")
                    (PRINT.STRING " to describe the memory for which Link will build")
                    (PRINT.STRING " image files. This portion must contain the MAP section.")
                    (PRINT.STRING " Every link file must have a SPECS section ")
                    (PRINT.STRING " and every SPECS section must have a MAP section.")
                    (PRINT.STRING " Other sections which can be specified in the SPECS section ")
                    (PRINT.STRING " include the TARGET and OVERLAY sections.")
                    (AM.NEW.LINE)
                    (PRINT.STRING "      The ROOT/ OVERLAY sections of the file describe which ")
                    (PRINT.STRING " objects to use to generate the memory images. The")
                    (PRINT.STRING
                          " structure of these statements is the same as the Link statement.")
                    (PRINT.STRING " The ROOT portion describes the root overlay and is")
                    (PRINT.STRING " required in all links. The Overlay portion is only")
                    (PRINT.STRING " permitted in those link files that contain an ")
                    (PRINT.STRING " overlay specification in their SPECS portion.")
                    (AM.NEW.LINE)
                    (PRINT.STRING "      For more detailed information on the Link file,")
                    (PRINT.STRING " please consult the Link or Link8051 User's guide.")))))
```

(AM.ASSIST.LINK.BASICS

```
  (LAMBOA NIL (BLOCK AM.ASSIST.LINK.BASICS (PRINT.STRING
                                   "      To execute a link, it is necessary to have several files."
                                        )
                    (PRINT.STRING " The list of required files includes the link file and the ")
                    (PRINT.STRING " files that are to be linked together.  The link file should")
                    (PRINT.STRING " contain all information needed to execute the link, as well as")
                    (PRINT.STRING " a list of files to be linked.")
                    (AM.NEW.LINE)
                    (PRINT.STRING
                          "      To see the format of the link file, please see the assistance")
                    (PRINT.STRING " under the Link format category, or see the Link or Link8051")
                    (PRINT.STRING " User's Guide.")))))
```

(AM.ASSIST.PSIZE.FORMAT

```
  (LAMBOA NIL                                    (IL:* IL:\; "Edited 9-Mar-89 12:06 by MAFoltman")
        (BLOCK AM.ASSIST.PSIZE.FORMAT (PRINT.STRING
                                    "      The psize specification is given in its own PSIZE "
                                        )
              (PRINT.STRING " section of the SPECS portion of the link file. It begins")
              (PRINT.STRING " with the statement")
              (PRINT.STRING "      PSIZE = ")
              (PRINT.STRING " and ends with the statement ")
              (PRINT.STRING "      ENO PSIZE")
              (PRINT.STRING " Between the two statements are the class/alias descriptions,")
              (PRINT.STRING " if any, followed by a series of one or more group descriptions.")
              (AM.NEW.LINE)
              (PRINT.STRING "      A class description name names a storage class for which")
              (PRINT.STRING " statistics will be produced. It has the format")
```

```
                (PRINT.STRING "      CLASS className")
                (PRINT.STRING " where className is the name of the class for which ")
                (PRINT.STRING " statistics are to be gathered.  This format will cause statistics")
                (PRINT.STRING " to be produced as a combined total for all overlays. If it is ")
                (PRINT.STRING " desired that the statistics will be broken down by overlay, the")
                (PRINT.STRING " class description would have the format")
                (PRINT.STRING "      OVERLAID CLASS className")
                (AM.NEW.LINE)
                (PRINT.STRING "      Further information on Alias grouping, overlaid aliases,  ")
                (PRINT.STRING " and grouping can be found in section 4.3 of the Link User's")
                (PRINT.STRING " Manual.")
                (AM.NEW.LINE))))
```

(AM.ASSIST.PSIZE.DEFN

```
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 9-Mar-89 11:54 by MAFoltman")
        (BLOCK AM.ASSIST.PSIZE.DEFN (PRINT.STRING
                                      "      The psize feature is used to produce statistics about "
                                     )
                (PRINT.STRING " memory usage within a system. The psize feature may be used ")
                (PRINT.STRING " with any Xerox supported 8085 based language (see Link manual,")
                (PRINT.STRING " section 4.3 for more details). The function is invoked by ")
                (PRINT.STRING " turning the z switch on and by providing the necessary psize ")
                (PRINT.STRING " specification.  The psize listing will be written to a file")
                (PRINT.STRING " with an extension of .psizeLst. (See Section 7.5 for more ")
                (PRINT.STRING " information on the psize listing file.)")
                (AM.NEW.LINE)
                (PRINT.STRING "      Statistics are produced or each module within the object")
                (PRINT.STRING " files for which psize information is requested.  Statistics may")
                (PRINT.STRING " be gathered for an individual storage class or for several  ")
                (PRINT.STRING " storage classes grouped together under a common alias.  ")
                (PRINT.STRING " Additionally, the statistics for a given class/alias may be")
                (PRINT.STRING " broken down by overlay or presented as a total for all overlays.")
                (AM.NEW.LINE)
                (PRINT.STRING "      For greater clarity, the files are arranged in groups.  The")
                (PRINT.STRING " files may be grouped as you see fit, e.g. by function, author, or")
                (PRINT.STRING " overlay.  Memory usage is subtotaled for each file and for each")
                (PRINT.STRING " group.")
                (AM.NEW.LINE)
                (PRINT.STRING "      For information on the format please see")
                (PRINT.STRING " AssistI Psize¬Psize Format or Section 4.3 of the Link User's guide.")
                (AM.NEW.LINE))))
```

(AM.ASSIST.PSIZE.SWITCHES

```
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 9-Mar-89 11:56 by MAFoltman")
        (BLOCK AM.ASSIST.PSIZE.SWITCHES (PRINT.STRING
                                          "      The z switch turns on the psize feature. It is only "
                                          )
                (PRINT.STRING " available for Link.")
                (AM.NEW.LINE))))
)
```

(IL:* IL:|;;| " ")

(IL:* IL:|;;| "Functions to bring up basic window")

```
(IL:DEFINEQ
```

(AMUSED.START

```
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 27-Feb-89 16:55 by MAFoltman")
```

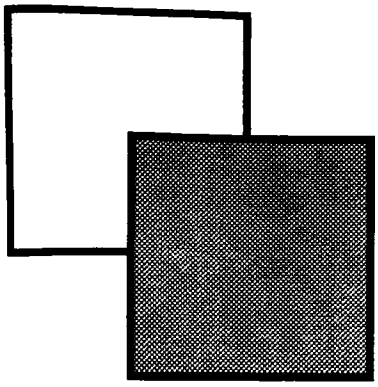
(IL:* IL:|;;| "Starting function for initial loads and ")
(IL:* IL:|;;| "starting up the Basic AMUSED window. Set up")
(IL:* IL:|;;| " the diagnose option freemenu and the context.")
(IL:* IL:|;;| "")

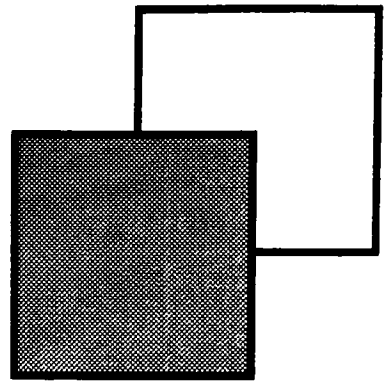
```
        (BLOCK AMUSED.START (PROG NIL
                                  (AM.DIAGNOSE.FREEMENU)
                                  (DEFINE.CONTEXT.BASE)
                                  (AMUSED.OPEN.BASIC.CMD.WINDOW)))))
```

```
(AMUSED.OPEN.BASIC.CMD.WINDOW
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 8-Mar-89 08:16 by MAFoltman")
        (BLOCK AMUSED.OPEN.BASIC.CMD.WINDOW
         (IL:* IL:|;;| "Open the main AMUSED window.")

              (SETQ AMUSED.MAIN.WINDOW (IL:CREATEW '(300 300 595 300) NIL))
              (IL:ATTACHMENU (SETQ MAIN.CMD.MENU (IL:|create| IL:MENU
                                                    IL:ITEMS IL:← '((DIAGNOSE! (
                                                                        AM.SETUP.DIAGNOSE
                                                                          ))
                                                                    (ASSIST! (AM.SETUP.ASSIST
                                                                          ))
                                                                    (DONE (AM.CLOSE.FILES))
                                                                    )
                                                    IL:TITLE IL:← 'COMMANDS
                                                    IL:MENUFONT IL:← (IL:FONTCREATE
                                                                       'MODERN 10 'BOLD)
                                                    IL:MENUCOLUMNS IL:← 3
                                                    IL:CENTERFLG IL:← T))
                    AMUSED.MAIN.WINDOW
                    'IL:TOP
                    'IL:JUSTIFY)
              (IL:ATTACHMENU (SETQ MAIN.TITLE.MENU (IL:|create| IL:MENU
                                                      IL:ITEMS IL:←
                                                      '(A MULTI-USER SOFTWARE ENVIRONMENT
                                                         DIAGNOSTIC)
                                                      IL:TITLE IL:← 'AMUSED
                                                      IL:MENUFONT IL:← (IL:FONTCREATE
                                                                         'MODERN 14 'BOLD)
                                                      IL:MENUTITLEFONT IL:← T
                                                      IL:CENTERFLG IL:← T))
                    AMUSED.MAIN.WINDOW
                    'IL:TOP
                    'IL:JUSTIFY))
        (IL:DSPSCROLL 'IL:ON AMUSED.MAIN.WINDOW)
        (IL:WINDOWPROP AMUSED.MAIN.WINDOW 'IL:ICONFN 'AMUSED.ICONFN)))

(AM.WINDOW.SCROLL
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 8-Mar-89 08:16 by MAFoltman")
        (BLOCK AM.WINDOW.SCROLL (PROG NIL
                                      (SETQ AMUSED.MAIN.WINDOW (CREATEW '(300 300 595 300) NIL))
                                      (WINDOWPROP AMUSED.MAIN.WINDOW 'SCROLLFN #'SCROLLBYREPAINTFN)
                                      (WINDOWPROP AMUSED.MAIN.WINDOW 'REPAINTFN #'
                                             REPAINT.AMUSED.WINDOW)
                                      (WINDOWPROP AMUSED.MAIN.WINDOW 'RESHAPEFN #'
                                             RESHAPE.AMUSED.WINDOW)
                                      (WINDOWPROP AMUSED.MAIN.WINDOW 'TEXT NIL)
                                      (RESHAPE.AMUSED.WINDOW AMUSED.MAIN.WINDOW)
                                      (RETURN AMUSED.MAIN.WINDOW)))))

(RESHAPE.AMUSED.WINDOW
  (LAMBDA (MYWINDOW)
        (BLOCK RESHAPE.AMUSED.WINDOW (IL:DSPFILL NIL NIL NIL MYWINDOW)
              (IL:WINDOWPROP MYWINDOW 'IL:ORIGX 0)
              (IL:WINDOWPROP MYWINDOW 'IL:ORIGY (- (IL:WINDOWPROP MYWINDOW 'IL:HEIGHT)
                                                   (IL:FONTPROP MYWINDOW 'IL:ASCENT)))
              (REPAINT.AMUSED.WINDOW MYWINDOW)
              (IL:WINDOWPROP MYWINDOW 'IL:EXTENT (IL:CREATEREGION 0 (+ (IL:DSPYPOSITION NIL
                                                                              MYWINDOW)
                                                                       (IL:FONTPROP MYWINDOW
                                                                              'IL:ASCENT))
                                                                  (IL:WINDOWPROP MYWINDOW 'IL:WIDTH)
                                                                  (- (IL:WINDOWPROP MYWINDOW 'IL:HEIGHT)
                                                                     (+ (IL:DSPYPOSITION NIL MYWINDOW)
                                                                        (IL:FONTPROP MYWINDOW 'IL:ASCENT)))))
              )))

(REPAINT.AMUSED.WINDOW
  (LAMBDA (WINDOW)
        (BLOCK REPAINT.AMUSED.WINDOW (IL:MOVETO (IL:WINDOWPROP WINDOW 'IL:ORIGX)
                                                (IL:WINDOWPROP WINDOW 'IL:ORIGY)
                                                WINDOW)
              (IL:PRINTDEF NIL 0 NIL NIL NIL WINDOW))))
```

```
(AM.CLOSE.FILES
  (LAMBDA NIL                                    (IL:* IL:\; "Edited 3-Apr-89 14:37 by MAFoltman")
        (BLOCK AM.CLOSE.FILES
         (IL:* IL:|;;| " Closes off log files and any open option windows. It")
         (IL:* IL:|;;| " prints a final message to the user. ")
         (IL:* IL:|;;| "   If the Log is set to true, then the file is made and the")
         (IL:* IL:|;;| " name is reported to the user.")
         (IL:* IL:|;;| "")

               (PROG NIL
                     (PRINT.STRING "Thank you for using AMUSED.")
                     (IL:CLOSEW DIAGNOSE.OPTIONS)
                     (COND
                        (ENABLE.LOG (SETQ AMUSED.LOGCOMS '((VARS LOG.FILE)))
                               (IL:MAKEFILE 'AMUSED.LOG)
                               (PRINC.STRING
                                 "The AMUSED activities log can be found under the name AMUSED.log "
                                    )
                               (SETQ LOG.FILE NIL)))))))

(AMUSED.ICONFN
  (LAMBDA (|window| |mouse|)                     (IL:* IL:\; "Edited 3-Apr-89 16:41 by MAFoltman")
        (BLOCK AMUSED.ICONFN (SETF AMUSEDICON (IL:ICONW AMUSEDBITS AMUSEDMASK)))))
)


(IL:* IL:|;;| "")


(IL:RPAQQ AMUSEDBITS #*(100 50)OOOOOOOOOOOOOOOOOOOOOOOOOO@@@H@@@@@@@@@@@@@@@@@@@@@@@A@@@KOOOOOOOOOOOOOOOOOOOO
OOM@@@J@@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@
@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@L@CH@ONOHO@OO
NOL@@@@@E@@@J@@L@ALALLGAOHGONGN@@@@@E@@@J@AN@ANCLLGCIHG@NGG@@@@@E@@@J@CO@ANCLLGC@LG@FGCH@@@@E@@@J@CO@AOGLLGC@
LG@FGAL@@@@E@@@J@GOHAOGLLGC@LG@BG@N@@@@E@@@J@GCHAOOLLGCANG@@G@F@GO@E@@@J@NCLAOOLLGCCOG@@G@G@OOHE@@@J@NALAKOLL
GC@@G@@G@GALALE@@@J@LANAKOLLGCH@G@@G@GCH@NE@@@JALANAIMLLGCL@GB@G@GCCFFE@@@JAH@NAIMLLGAN@GN@G@GFBDCE@@@JCH@OAH
ILLG@O@GN@G@GFCFCE@@@JCH@OAHILLG@GHGB@G@GF@@CE@@@JC@@OAHALLG@CLG@@G@GFOACE@@@JCOOOAHALLG@ANG@@G@GFBBCE@@@JCOO
OAHALLG@@NG@@G@GCALFE@@@JC@@OAHALLG@@NG@@G@GCH@NE@@@JC@@OAHALLG@@NG@@G@GALALE@@@JC@@OAHALLG@@NG@@G@G@OOHE@@@J
C@@OAHALLGGLNG@@G@G@GO@E@@@JC@@OAHALLGGHNG@@G@G@@@@E@@@JC@@OAHALLGC@NG@BG@G@@@@E@@@JC@@OAHALLGC@NG@FG@F@@@@E@
@@JC@@OAHALLGC@NG@FG@N@@@@E@@@JC@@OAHALLGCILG@NGAL@@@@E@@@JC@AOIHCLNOAOHGONGOH@@@@E@@@JGHCOOLGNGN@O@OONOO@@@@
@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@
@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@@E@@@J@@@@@@@@@@@@@@@@@
@@@@@@@E@@@J@@@@@@@@@@@@@@@@@@@@@@@E@@@KOOOOOOOOOOOOOOOOOOOOOOOM@@@H@@@@@@@@@@@@@@@@@@@@@@@A@@@OOOOOOOOOOOOOOO
OOOOOOOOOOO@@@
)

(IL:RPAQQ AMUSEDMASK #*(100 50)OOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOO
OOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
@@@OOOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOO
OOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOO
OOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOOOOOOOOOO@@@OOOOOOOOOOOOOOOOOO
OOOOOOOOOOOO@@@
)


(IL:* IL:|;;|
"(ADDVARS (BACKGROUNDMENUCOMMANDS (\"AMUSED\" '(XCL::AMUSED.START) \"Opens the main window
for AMUSED diagnostics\")))"
)


(IL:* IL:|;;| "")
```

```
(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA
             AMUSED.ICONFN AM.CLOSE.FILES REPAINT.AMUSED.WINDOW RESHAPE.AMUSED.WINDOW
                   AM.WINDOW.SCROLL AMUSED.OPEN.BASIC.CMD.WINDOW AMUSED.START
                   AM.ASSIST.PSIZE.SWITCHES AM.ASSIST.PSIZE.DEFN AM.ASSIST.PSIZE.FORMAT
                   AM.ASSIST.LINK.BASICS AM.ASSIST.LINK.FORMAT AM.ASSIST.SEARCH.PATH.RESET
                   AM.ASSIST.SEARCH.PATH.ACTIVE AM.ASSIST.USERCM.SETUP AM.ASSIST.USERCM.OPTIONS
                   AM.ASSIST.VALID.SWITCHES AM.ASSIST.INVALID.SWITCHES AM.SETUP.ASSIST
                   AM.REPEAT.DIAGNOSE AM.DIAGNOSE.MODE AM.SETUP.DIAGNOSE)
)
(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL (3824 7513 (AM.SETUP.DIAGNOSE 3837 . 4547) (AM.DIAGNOSE.MODE 4549 . 6832) (
AM.REPEAT.DIAGNOSE 6834 . 7511)) (7582 11469 (AM.SETUP.ASSIST 7595 . 11467)) (11565 39299 (
AM.ASSIST.INVALID.SWITCHES 11578 . 13007) (AM.ASSIST.VALID.SWITCHES 13009 . 22269) (
AM.ASSIST.USERCM.OPTIONS 22271 . 23170) (AM.ASSIST.USERCM.SETUP 23172 . 28686) (
AM.ASSIST.SEARCH.PATH.ACTIVE 28688 . 29654) (AM.ASSIST.SEARCH.PATH.RESET 29656 . 30761) (
AM.ASSIST.LINK.FORMAT 30763 . 33830) (AM.ASSIST.LINK.BASICS 33832 . 34763) (AM.ASSIST.PSIZE.FORMAT
34765 . 36676) (AM.ASSIST.PSIZE.DEFN 36678 . 38866) (AM.ASSIST.PSIZE.SWITCHES 38868 . 39297)) (39387
46277 (AMUSED.START 39400 . 40009) (AMUSED.OPEN.BASIC.CMD.WINDOW 40011 . 42591) (AM.WINDOW.SCROLL
42593 . 43505) (RESHAPE.AMUSED.WINDOW 43507 . 44690) (REPAINT.AMUSED.WINDOW 44692 . 45008) (
AM.CLOSE.FILES 45010 . 46066) (AMUSED.ICONFN 46068 . 46275)))))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>TMYCINTOOL.;19

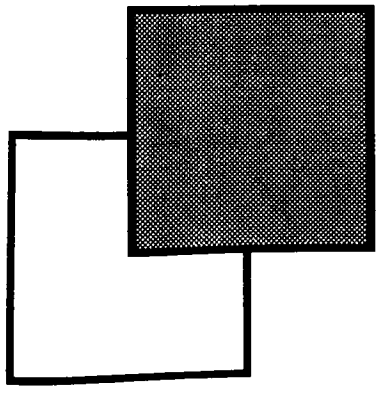
Created 1-May-89 8:46:54

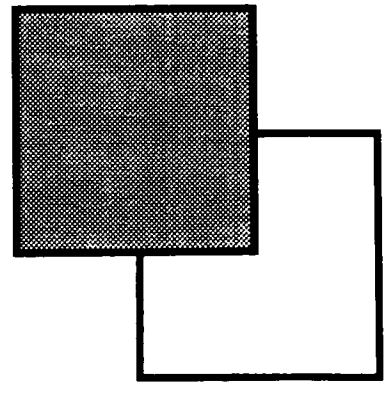
Printed 1-May-89 8:48:39

19 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 2 more





```
(OEFINE-FILE-INFO ▮PACKAGE "XCL-USER" ▮READTABLE "XCL" ▮BASE 10)
(IL:FILECREATEO " 5-Apr-89 11:27:13" IL:{DSK}<LISPFILES>THESIS>CODE>TMYCINTOOL.\;19 60889

      IL:|changes| IL:|to:|  (IL:VARS IL:TMYCINTOOLCOMS)
                             (IL:FUNCTIONS SET.UP.HISTORY.LIST AOO.TO.HISTORY.LIST)

      IL:|previous| IL:|date:| "17-Mar-89 16:02:22" IL:{OSK}<LISPFILES>THESIS>COOE>TMYCINTOOL.\;17)


(IL:PRETTYCOMPRINT IL:TMYCINTOOLCOMS)

(IL:RPAQQ IL:TMYCINTOOLCOMS
          ((IL:P (IN-PACKAGE 'IL:XCL-USER))
           (IL:PROP (IL:MAKEFILE-ENVIRONMENT)
                  IL:TMYCINTOOL)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| "     Variable descriptions")


(IL:* IL:|;;;| " ALLRULES is for the user. It is  not used by the system. ")


(IL:* IL:|;;;| " CNTXT is the current data context.")


(IL:* IL:|;;;| "  CNTXTNAME is the name of the top context.  ")


(IL:* IL:|;;;| " TALLY is for CF (CERTAINTY FACTOR) calculations. ")


(IL:* IL:|;;;| " PREVGOALS is for loop prevention in  BC-GOAL. ")


(IL:* IL:|;;;| " RUNRULERULENAME is used in answering the why questions")


(IL:* IL:|;;;| ".")

            (IL:VARIABLES CNTXTNAME PREVGOALS RUNRULERULENAME TALLY ALLRULES CNTXT)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| "Functions used in the base TMYCIN tool.")


(IL:* IL:|;;;| ".")


(IL:* IL:|;;;| "  Functions for premise clause preparation")

            (IL:FUNCTIONS $ANO $ANOEXPR $OR $OREXPR KNOWN KNOWNEXPR SAME SAMEEXPR OO-ALL CONCLUOE
                   NOTSAME NOTSAMEEXPR VAL1 VAL1EXPR)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| " Set up functions for creating context and rules")

            (IL:FUNCTIONS OEFCONTEXT OEFRULE OEFRULES)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| " Functions for maintaining a history list for new rules as needed.")


(IL:* IL:|;;;| " Functions to supply the learning to TMYCIN.")
```

```
(IL:* IL:|;;;| "")

            (IL:FUNCTIONS SET.UP.HISTORY.LIST SET.UP.NEW.RULE ADD.TO.HISTORY.LIST)

(IL:* IL:|;;;| "Interaction with user")

            (IL:FUNCTIONS DOCONSULT ASKUSER GETINITIALDATA)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| " Internal functions for processing")

            (IL:FUNCTIONS BC-GOAL CFCOMBINE CLAUSEPARM CNTXTGET CONCLUDEEXPR FINDPARMCTX FINDPARMOES
                   GLMKATOM MAKE-CONTEXT PARMGET RUNRULE)

(IL:* IL:|;;;| "")


(IL:* IL:|;;;| "EMYCIN-style comparisons")

            (IL:FUNCTIONS GREATEQ* GREATERP* LESSEQ* LESSP* BETWEEN*)

(IL:* IL:|;;;;| "")


(IL:* IL:|;;;;| " Printing functions,")

            (IL:FUNCTIONS ENGLRULE PRINTCLAUSE PRINTCONC PRINTCONCLUSION PRINTPREMISES CLSPACES
                   SHOWPROPS SHOWRULE)

(IL:* IL:|;;;;| "")


(IL:* IL:|;;;;| " Questioning functions, Reasoning behind tool")

            (IL:FUNCTIONS WHY WHYEXPR WHYNOT WHYNOTEXPR)))
(IN-PACKAGE 'IL:XCL-USER)

(IL:PUTPROPS IL:TMYCINTOOL IL:MAKEFILE-ENVIRONMENT (:PACKAGE "XCL-USER" :READTABLE "XCL" :BASE 10
                                                                    ))


(IL:* IL:|;;;| "")


(IL:* IL:|;;;| "      Variable descriptions")


(IL:* IL:|;;;| " ALLRULES is for the user. It is not used by the system. ")


(IL:* IL:|;;;| " CNTXT is the current data context.")


(IL:* IL:|;;;| " CNTXTNAME is the name of the top context. ")


(IL:* IL:|;;;| " TALLY is for CF (CERTAINTY FACTOR) calculations. ")
```

(IL:* IL:|;;;| " PREVGOALS is for loop prevention in BC-GOAL. ")

(IL:* IL:|;;;| " RUNRULERULENAME is used in answering the why questions")

(IL:* IL:|;;;| ".")

```
(DEFVAR CNTXTNAME NIL)

(DEFVAR PREVGOALS NIL)

(DEFVAR RUNRULERULENAME NIL)

(DEFVAR TALLY 0)

(DEFVAR ALLRULES NIL)

(DEFVAR CNTXT NIL)
```

(IL:* IL:|;;;| "")

(IL:* IL:|;;;| "Functions used in the base TMYCIN tool.")

(IL:* IL:|;;;| ".")

(IL:* IL:|;;;| " Functions for premise clause preparation")

```
(DEFMACRO $AND (&REST CONDITIONS)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "AND together clauses so long as CF > .2, return minimum CF.")
(IL:* IL:|;;;;| "")
                                        '($ANDEXPR ',CONDITIONS))

(DEFUN $ANDEXPR (CLAUSES)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;|
        "AND together clauses so long as CF > .2, return minimum CF to macro call $AND.")
          (IL:* IL:|;;| "")

                                (PROG (RESULT CR)
                                  LP  (COND
                                         ((NULL CLAUSES)
                                          (RETURN RESULT)))
                                      (SETQ CR (EVAL (CAR CLAUSES)))
                                      (SETQ CLAUSES (CDR CLAUSES))
                                      (COND
                                         ((NULL CR)
                                          (RETURN NIL))
```

```
                                    ((NOT (NUMBERP CR))
                                     (AM.ERROR "Bad result from antecedent clause"))
                                    ((<= CR 0.2)
                                     (RETURN NIL)))
                                (SETQ RESULT (COND
                                                ((NULL RESULT)
                                                 CR)
                                                (T (MIN RESULT CR))))
                                (GO LP)))


(DEFMACRO $OR (&REST CONDITIONS)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "OR together clauses, return maximum CF, stop on CF = 1.0.")
(IL:* IL:|;;;;| "")

                                    '($OREXPR ',CONDITIONS))


(DEFUN $OREXPR (CLAUSES)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;|
"OR together clauses, return maximum CF, stop on CF = 1.0. Return CF to macro call $OR.")
(IL:* IL:|;;;;| "")

                                (PROG (RESULT CR)
                                  LP  (COND
                                         ((NULL CLAUSES)
                                          (RETURN RESULT)))
                                      (SETQ CR (EVAL (CAR CLAUSES)))
                                      (SETQ CLAUSES (CDR CLAUSES))
                                      (COND
                                         ((NULL CR))
                                         ((NOT (NUMBERP CR))
                                          (AM.ERROR "Bad result from antecedent clause")))
                                      (SETQ RESULT (COND
                                                      ((NULL RESULT)
                                                       CR)
                                                      (T (MAX RESULT CR))))
                                      (COND
                                         ((EQUAL RESULT 1.0)
                                          (RETURN RESULT)))
                                      (GO LP)))


(OEFMACRO KNOWN (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Test for a specified value with CF > .2, return 1.0")
(IL:* IL:|;;;;| "")

                                    '(KNOWNEXPR ,CNTXT ',PARM ',VALUE))


(DEFUN KNOWNEXPR (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;|
"Actual test for a specified value with CF >.2, returns 1.0 to the macro call KNOWN.")
(IL:* IL:|;;;;| " ")

                                     (PROG (VALS PAIR)
                                           (SETQ VALS (PARMGET CNTXT PARM))
                                           (RETURN (AND (SETQ PAIR (ASSOC VALUE VALS))
                                                        (> (CADR PAIR)
                                                           0.2)
                                                        1.0))))


(DEFMACRO SAME (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Test for a specified value with CF > .2, returns the CF ")
(IL:* IL:|;;;;| "")

                                    '(SAMEEXPR ,CNTXT ',PARM ',VALUE))
```

```
(DEFUN SAMEEXPR (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;|
"Actual text for a specified value with CF > .2, it returns the CF to the macro call SAME.")
(IL:* IL:|;;;;| "")

                                        (PROG (VALS PAIR)
                                              (SETQ VALS (PARMGET CNTXT PARM))
                                              (RETURN (AND (SETQ PAIR (ASSOC VALUE VALS))
                                                           (> (CADR PAIR)
                                                              0.2)
                                                           (CADR PAIR)))))


(DEFMACRO DO-ALL (&REST LVARFORDOALL)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Do a set of things in consequent")
(IL:* IL:|;;;;| "")

                                           '(MAPC #'EVAL ',LVARFORDOALL))


(OEFMACRO CONCLUDE (CNTXT PARM VALUE TALLY RULECF)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Basic rule conclusion function")
(IL:* IL:|;;;;| "    (CONCLUDE CNTXT parm val TALLY cf)")
(IL:* IL:|;;;;| "")

                                                      '(CONCLUDEEXPR ,CNTXT ',PARM
                                                             ',VALUE
                                                             ,TALLY
                                                             ,RULECF))


(DEFMACRO NOTSAME (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Test for a specified value with CF < = .2, return 1.0")
(IL:* IL:|;;;;| "")

                                          '(NOTSAMEEXPR ,CNTXT ',PARM ',VALUE))


(DEFUN NOTSAMEEXPR (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;|
" Actual test for a specified value with CF < = .2, returns 1.0 to the macro call, NOTSAME.")
(IL:* IL:|;;;;| "")

                                           (PROG (VALS PAIR)
                                                 (SETQ VALS (PARMGET CNTXT PARM))
                                                 (RETURN (COND
                                                           ((OR (NULL (SETQ PAIR (ASSOC VALUE VALS)))
                                                                (<= (CADR PAIR)
                                                                 0.2))
                                                            1.0)
                                                           (T NIL)))))


(DEFMACRO VAL1 (CNTXT PARM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Get the value of a parameter. The value with the highest CF is returned.")
(IL:* IL:|;;;;| "")

                                       '(VAL1EXPR ,CNTXT ',PARM))


(DEFUN VAL1EXPR (CNTXT PARM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Actual function to get the value of a parameter. The value with the highest CF is returned to the macro call VAL1.")
(IL:* IL:|;;;;| "")

                                        (PROG (PARMVALS MAXPAIR)
                                              (SETQ PARMVALS (PARMGET CNTXT PARM))
```

```
                                    LP  (COND
                                          ((NULL PARMVALS)
                                           (RETURN (AND MAXPAIR (CAR MAXPAIR))))
                                          ((OR (NULL MAXPAIR)
                                               (> (CADAR PARMVALS)
                                                  (CAOR MAXPAIR)))
                                           (SETQ MAXPAIR (CAR PARMVALS))))
                                        (SETQ PARMVALS (CDR PARMVALS))
                                        (GO LP)))
```

(IL:* IL:|;;;| "")

(IL:* IL:|;;;| " Set up functions for creating context and rules")

```
(DEFUN DEFCONTEXT (CONTEXT PARMS INITIALDATA GOALS)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;|
        "Define a context class. ARgs are context name, parameters, intialdata, goals")
          (IL:* IL:|;;|
        " Each parameter specification is a list of the parameter name, expected type,")
          (IL:* IL:|;;|
        " and reprompt string. The latter two are used mainly in response to \"?\" input.")
          (IL:* IL:|;;| " An expected type of NIL is used for YES/NO parameters.")
          (IL:* IL:|;;| "E.g., ")
          (IL:* IL:|;;| "   (DEFCONTEXT 'PATIENT")
          (IL:* IL:|;;| "          '((NAME STRING)")
          (IL:* IL:|;;|
        "          (AGE POSNUMB \"What is the age of patient in years?\")")
          (IL:* IL:|;;| "          (RASH NIL \"Is there an obvious rash?\")")
          (IL:* IL:|;;|
        "          (TEMPERATURE POSNUMB \"What is the temp. in degrees F?\")")
          (IL:* IL:|;;| "          (DISEASE (MEASLES RUBELLA PIMPLES)))")
          (IL:* IL:|;;| "          '(NAME AGE)")
          (IL:* IL:|;;| "          '(DISEASE))")
          (IL:* IL:|;;| "")

                                                  (PROG NIL
                                                        (SETQ CNTXTNAME CONTEXT)
                                                        (SETF (GET CONTEXT 'PARAMETERS)
                                                              PARMS)
                                                        (SETF (GET CONTEXT 'INITIALDATA)
                                                              INITIALDATA)
                                                        (SETF (GET CONTEXT 'GOALS)
                                                              GOALS)))


(DEFUN DEFRULE (RULE)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;| "Define a new rule, index on conclusions, add to ALLRULES.")
          (IL:* IL:|;;| " A rule looks like:")
          (IL:* IL:|;;| "   (RULE101 ($AND (...) ...) (CONCLUDE ...))")
          (IL:* IL:|;;| "")

                              (PROG (PARMS CONCPART RULENAME)
                                    (SETQ RULENAME (COND
                                                       ((EQ (CAR RULE)
                                                            'IF)
                                                        (GLMKATOM 'RULE))
                                                       (T (CAR RULE))))
                                    (SETF (GET RULENAME 'RULE)
                                          RULE)
                                    (SETQ CONCPART (CADDR RULE))
                                    (COND
                                       ((EQ (CAR CONCPART)
                                            'CONCLUDE)
                                        (SETQ PARMS (LIST (CADDR CONCPART))))
                                       ((EQ (CAR CONCPART)
```

```
                                    'DO-ALL)
                                (SETQ PARMS (MAPCAN #'(LAMBDA (CONC)
                                                        (COND
                                                           ((EQ (CAR CONC)
                                                                'CONCLUDE)
                                                            (LIST (CADDR CONC)))))
                                                    (CDR CONCPART))))
                               (T (AM.ERROR "Incorrect conclusion part of rule")))
                            (MAPC #'(LAMBDA (PARM)
                                        (OR (MEMBER RULENAME (GET PARM 'RULES))
                                            (SETF (GET PARM 'RULES)
                                                  (CONS RULENAME (GET PARM 'RULES)))))) PARMS)
                            (OR (MEMBER RULENAME ALLRULES)
                                (SETQ ALLRULES (NCONC ALLRULES (CONS RULENAME NIL))))))

(DEFMACRO DEFRULES (&REST LVARFORDEFRULES)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Define a set of rules")
(IL:* IL:|;;;;| "")

                                        '(AND (MAPC #'DEFRULE ',LVARFORDEFRULES)
                                              T))



(IL:* IL:|;;;;| "")



(IL:* IL:|;;;;| " Functions for maintaining a history list for new rules as needed.")



(IL:* IL:|;;;;| " Functions to supply the learning to TMYCIN.")



(IL:* IL:|;;;;| "")

(DEFUN SET.UP.HISTORY.LIST NIL (SETQ HISTORY.LIST NIL)
                               (SETF (GET HISTORY.LIST 'PREMISES)
                                     NIL)
                               (SETF (GET HISTORY.LIST 'CONCLUSION)
                                     NIL)
                               (SETQ NEW.RULE.NUMBER 0)
                               (SETQ GLOBAL.REGISTERED NIL)
                               (SETQ NEW.RULES.LIST '(DEFRULES)))

(DEFUN SET.UP.NEW.RULE (CNTXT) (PROG (NEW.RULE.NAME PREM NEW.PREMISES ALL.PREMISES ALL.CONDITIONS
                                      CONCLUSION)
                                   (SETQ NEW.RULE.NUMBER (1+ NEW.RULE.NUMBER))
                                   (SETQ NEW.RULE.NAME (READ-FROM-STRING (IL:CONCAT
                                                                          'RULE00
                                                                          NEW.RULE.NUMBER)))
                                   (DOLIST (PREM (GET HISTORY.LIST 'PREMISES))
                                          (COND
                                             ((IL:ASSOC (CAR PREM)
                                                    (GET CNTXT 'PARAMETERS))
                                              (COND
                                                 ((EQUAL (CADR PREM)
                                                         'YES)
                                                  (SETQ NEW.PREMISES
                                                        (APPEND NEW.PREMISES
                                                                (LIST (LIST 'SAME 'CNTXT
                                                                            (CAR PREM)
                                                                            (CADR PREM))))))
                                                 ((EQUAL (CADR PREM 'NO))
```

```
                                                (SETQ NEW.PREMISES
                                                      (APPEND NEW.PREMISES
                                                              (LIST (LIST 'NOTSAME
                                                                          'CNTXT
                                                                          (CAR PREM)
                                                                          (CADR PREM))))))))
                                                   (T (SETQ NEW.PREMISES (APPEND NEW.PREMISES
                                                                                 (LIST PREM)))))
                                             NEW.PREMISES)
                                       (SETQ ALL.CONDITIONS (APPEND '($AND) NEW.PREMISES))
                                       (SETQ CONCLUSION (LIST 'CONCLUDE 'CNTXT
                                                              (CAAR (GET HISTORY.LIST 'CONCLUSION))
                                                              (CADAR (GET HISTORY.LIST 'CONCLUSION)
                                                                     )
                                                              'TALLY
                                                              (* 1000 (CADDAR (GET HISTORY.LIST
                                                                                   'CONCLUSION)))))
                                       (SETQ NEW.RULES.LIST (APPEND NEW.RULES.LIST
                                                                    (LIST (LIST NEW.RULE.NAME
                                                                                ALL.CONDITIONS
                                                                                CONCLUSION))))
                                       (SETQ IL:NEWRULESCOMS '((IL:VARS NEW.RULES.LIST)))
                                       (IL:MAKEFILE 'NEWRULES)))

(DEFUN ADD.TO.HISTORY.LIST (RESULT)
   (PROG NIL
         (COND
            ((NOT GLOBAL.REGISTERED)
             (SETQ GLOBAL.REGISTERED T)
             (COND
                ((EQUAL LINKER.USED 'LINK)
                 (SETF (GET HISTORY.LIST 'PREMISES)
                       (APPEND (GET HISTORY.LIST 'PREMISES)
                               (LIST (LIST 'SAME.GLOBAL 'LINKER.USED ''LINK)))))
                (T (SETF (GET HISTORY.LIST 'PREMISES)
                         (APPEND (GET HISTORY.LIST 'PREMISES)
                                 (LIST (LIST 'SAME.GLOBAL 'LINKER.USED ''LINK8051))))))
             (COND
                ((EQUAL ERROR.TYPE 'WARNING)
                 (SETF (GET HISTORY.LIST 'PREMISES)
                       (APPEND (GET HISTORY.LIST 'PREMISES)
                               (LIST (LIST 'SAME.GLOBAL 'ERROR.TYPE ''WARNING)))))
                ((EQUAL ERROR.TYPE 'ERROR)
                 (SETF (GET HISTORY.LIST 'PREMISES)
                       (APPEND (GET HISTORY.LIST 'PREMISES)
                               (LIST (LIST 'SAME.GLOBAL 'ERROR.TYPE ''ERROR)))))
                (T (SETF (GET HISTORY.LIST 'PREMISES)
                         (APPEND (GET HISTORY.LIST 'PREMISES)
                                 (LIST (LIST 'SAME.GLOBAL 'ERROR.TYPE ''FATAL))))))))
         (COND
            ((LISTP RESULT)
             (SETF (GET HISTORY.LIST 'PREMISES)
                   (APPEND (GET HISTORY.LIST 'PREMISES)
                           (LIST (LIST 'SAME.GLOBAL (LIST 'SPECIFIC.QUESTION (LIST 'QUOTE (CAR RESULT)
                                                                                   )
                                                                                   (CADR RESULT))
                                       (LIST 'QUOTE (CADOR RESULT)))))))))))
```

(IL:* IL:|;;;| "Interaction with user")

```
(DEFUN DOCONSULT (TOPCLASS)
       (IL:* IL:|;;;| "")
       (IL:* IL:|;;;| " Run a consultation. E.G. (DOCONSULT 'ROCK)")
       (IL:* IL:|;;;| "")

                          (PROG (TOPCTX GOALS)
                                (AM.NEW.LINE)
                                (AM.NEW.LINE)
                                (SET.UP.HISTORY.LIST)
```

```
                                    (SETQ GOALS (GET TOPCLASS 'GOALS))
                                    (SETQ TOPCTX (MAKE-CONTEXT TOPCLASS NIL))
                                    (GETINITIALDATA TOPCTX)
                                    (MAPC #'(LAMBDA (GOAL)
                                                    (BC-GOAL TOPCTX GOAL NIL)) GOALS)
                                    (AM.NEW.LINE)
                                    (AM.NEW.LINE)
                                    (PRINC.STRING "The conclusions for ")
                                    (PRINC.STRING TOPCTX)
                                    (PRINC.STRING " are as follows:")
                                    (AM.NEW.LINE)
                                    (MAPC #'(LAMBDA (GOAL)
                                                    (PRINTCONC GOAL)
                                                    (PRINTCONCLUSION TOPCTX GOAL)) GOALS)
                                    (COND
                                       ((NOT (NULL (GET HISTORY.LIST 'CONCLUSION)))
                                        (SET.UP.NEW.RULE TOPCLASS)))))


(DEFUN ASKUSER (CNTXT PARM)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;| " Ask the user for a data value")
          (IL:* IL:|;;| "")

                                  (PROG (INP PARMDES NEED.INPUT?)
                                        (SETQ PARMDES (FINDPARMDES CNTXT PARM))
                                    TOP (COND
                                           ((NULL (CAOR PARMDES))
                                            (SETQ INP (CREATE.MENU (IL:CONCAT "Is " CNTXT " " PARM "? "
                                                                              )
                                                                   '(YES NO ?) T))
                                            (AM.NEW.LINE)
                                            (SETQ NEED.INPUT? NIL))
                                           (T (SETQ NEED.INPUT? T)
                                              (AM.NEW.LINE)
                                              (PRINC.STRING "What is the ")
                                              (PRINC.STRING PARM)
                                              (PRINC.STRING " of ")
                                              (PRINC.STRING CNTXT)
                                              (PRINC.STRING "? "))))
                                    LP  (AM.NEW.LINE)
                                        (COND
                                           (NEED.INPUT? (SETQ INP (AM.GET.INPUT.NUMBER))))
                                        (SETQ NEED.INPUT? T)
                                        (COND
                                           ((EQ INP '?)
                                            (COND
                                               ((CDDR PARMDES)
                                                (PRINC.STRING (CADDR PARMDES))
                                                (AM.NEW.LINE)))
                                            (PRINC.STRING "Expected values are: ")
                                            (PRINC.STRING (CADR PARMDES))
                                            (AM.NEW.LINE)
                                            (GO LP))
                                           ((EQ INP 'WHY)
                                            (AM.NEW.LINE)
                                            (PRINC.STRING "We are examining the following rule:")
                                            (AM.NEW.LINE)
                                            (AM.NEW.LINE)
                                            (SHOWRULE RUNRULERULENAME)
                                            (AM.NEW.LINE)
                                            (GO TOP))
                                           ((MEMBER INP '(Y YES N NO))
                                            (SETQ PARMDES (FINDPARMDES CNTXT PARM))
                                            (COND
                                               ((NULL (CADR PARMDES))
                                                (SETQ INP (COND
                                                             ((MEMBER INP '(N NO))
                                                              (LIST 'YES -1.0))
                                                             (T (LIST 'YES 1.0))))))))
                                        (COND
                                           ((NOT (EQ PARM 'SOLUTION))
                                            (SETF (GET HISTORY.LIST 'PREMISES)
                                                  (APPEND (GET HISTORY.LIST 'PREMISES)
```

```
                                                    (LIST (LIST PARM (CAR INP))))))
                                          (T (SETF (GET HISTORY.LIST 'CONCLUSION)
                                                   (APPEND (GET HISTORY.LIST 'CONCLUSION)
                                                           (LIST (LIST PARM INP 1.0))))))
                                    (RETURN (COND
                                              ((AND (CONSP INP)
                                                    (CONSP (CAR INP)))
                                               INP)
                                              ((CONSP INP)
                                               (LIST INP))
                                              (T (LIST (LIST INP 1.0)))))))

(DEFUN GETINITIALDATA (CNTXT)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Get initial data for a class.")
(IL:* IL:|;;;;| "")

                                  (PROG (CLASS)
                                        (SETQ CLASS (GET CNTXT 'ISA))
                                        (MAPC #'(LAMBDA (PARM)
                                                        (SETF (GET CNTXT PARM)
                                                              (ASKUSER CNTXT PARM)))
                                              (GET CLASS 'INITIALDATA))))
```

(IL:* IL:|;;;| "")

(IL:* IL:|;;;| " Internal functions for processing")

```
(DEFUN BC-GOAL (CNTXT PARM PREVGOALS)
           (IL:* IL:|;;| "")
           (IL:* IL:|;;| " Backchain through rules to try to conclude goals.")
           (IL:* IL:|;;| "")

                                           (PROG (CNTXTP RULES ASKED)
                                                 (SETQ CNTXTP (FINDPARMCTX CNTXT PARM))
           (IL:* IL:|;;| "If parm is already traced, don't do it again.")

                                                 (COND
                                                    ((GET CNTXTP PARM))
                                                    ((OR (GET PARM 'ASKFIRST)
                                                         (NULL (SETQ RULES (GET PARM 'RULES))))
                                                     (SETQ ASKED T)
                                                     (SETF (GET CNTXTP PARM)
                                                           (ASKUSER CNTXTP PARM)))
                                                    (T (MAPC #'(LAMBDA (RULE)
                                                                       (RUNRULE CNTXT RULE)) RULES)))
                                                 (COND
                                                    ((GET CNTXTP PARM))
                                                    ((NOT ASKED)
                                                     (SETF (GET CNTXTP PARM)
                                                           (ASKUSER CNTXTP PARM)))
                                                    (T (SETF (GET CNTXTP PARM)
                                                             (COPY-TREE '((NOVALUE 0.0))))))
                                                 (RETURN (GET CNTXTP PARM))))


(DEFUN CFCOMBINE (OLD NEW)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "EMYCIN CF calculation algorithm.")
(IL:* IL:|;;;;| "\"It ain't perfect, but it's better than its inputs usually are.\"")
(IL:* IL:|;;;;| "")

                                  (COND
                                     ((NULL OLD)
                                      NEW)
                                     ((NULL NEW)
```

```
                                    OLD)
                                   ((NOT (AND (NUMBERP OLD)
                                              (NUMBERP NEW)))
                                    (AM.ERROR "Bad CF"))
                                   ((OR (AND (EQUAL OLD 1.0)
                                             (EQUAL NEW -1.0))
                                        (AND (EQUAL OLD -1.0)
                                             (EQUAL NEW 1.0)))
                                    (AM.ERROR "Contradiction in CF values"))
                                   ((OR (EQUAL OLD 1.0)
                                        (EQUAL NEW 1.0))
                                    1.0)
                                   ((OR (EQUAL OLD -1.0)
                                        (EQUAL NEW -1.0))
                                    -1.0)
                                   ((AND (MINUSP OLD)
                                         (MINUSP NEW))
                                    (- (CFCOMBINE (- OLD)
                                                  (- NEW))))
                                   ((AND (NOT (MINUSP OLD))
                                         (NOT (MINUSP NEW)))
                                    (+ OLD (* NEW (- 1.0 OLD))))
                                   (T (IL:FQUOTIENT (+ OLD NEW)
                                                    (- 1.0 (MIN (ABS OLD)
                                                                (ABS NEW)))))))

(DEFUN CLAUSEPARM (CLAUSE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Return the parameter name for a clause")
(IL:* IL:|;;;;| "")

                                  (COND
                                     ((MEMBER (CAR CLAUSE)
                                              '(SAME NOTSAME KNOWN))
                                      (CADDR CLAUSE))
                                     ((MEMBER (CAR CLAUSE)
                                              '(GREATERP* GREATEQ* LESSP* LESSEQ* BETWEEN*))
                                      (CADDR (CADR CLAUSE)))
                                     (T '???)))


(DEFUN CNTXTGET (CNTXT PARM)
(IL:* IL:|;;;;| " ")
(IL:* IL:|;;;;| "Get the value of a parameter from a context, looking up parent chain.")
(IL:* IL:|;;;;| "")

                                   (COND
                                      ((NULL CNTXT)
                                       NIL)
                                      ((GET CNTXT PARM))
                                      (T (CNTXTGET (GET CNTXT 'PARENT)
                                                   PARM))))


(DEFUN CONCLUDEEXPR (CNTXT PARM VAL TALLY RULECF)

(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Basic rule conclusion function, called from macro CONCLUDE.")
(IL:* IL:|;;;;| "    (CONCLUDE CNTXT parm val TALLY cf)")
(IL:* IL:|;;;;| "")

   (PROG (CNTXTP VALS PAIR NEWCF)
         (SETQ NEWCF (* TALLY (IL:FQUOTIENT (FLOAT RULECF)
                                            1000.0)))
         (SETQ CNTXTP (FINDPARMCTX CNTXT PARM))
         (SETQ VALS (GET CNTXTP PARM))
         (COND
            ((SETQ PAIR (ASSOC VAL VALS))
             (RPLACA (CDR PAIR)
                     (CFCOMBINE (CADR PAIR)
                                NEWCF)))
            (T (SETF (GET CNTXTP PARM)
                     (NCONC VALS (LIST (LIST VAL NEWCF))))))))
```

```
(DEFUN FINDPARMCTX (CNTXT PARM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Find proper level of context for a parameter")
(IL:* IL:|;;;;| "")

                                    (PROG (CLASS)
                                          (COND
                                             ((NULL CNTXT)
                                              (AM.ERROR (LIST "Couldnt find context for parm" PARM
                                                              "probably left out of defcontext")))
                                             ((NOT (SETQ CLASS (GET CNTXT 'ISA)))
                                              (AM.ERROR "Context has no Class")))
                                          (RETURN (COND
                                                    ((ASSOC PARM (GET CLASS 'PARAMETERS))
                                                     CNTXT)
                                                    (T (FINDPARMCTX (GET CNTXT 'PARENT)
                                                                    PARM))))))

(DEFUN FINDPARMDES (CNTXT PARM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Find description for a parameter")
(IL:* IL:|;;;;| "")

                                    (PROG (CLASS)
                                          (COND
                                             ((NULL CNTXT)
                                              (AM.ERROR "Couldn't find context for parm"))
                                             ((NOT (SETQ CLASS (GET CNTXT 'ISA)))
                                              (AM.ERROR "Context has no Class")))
                                          (RETURN (COND
                                                    ((ASSOC PARM (GET CLASS 'PARAMETERS)))
                                                    (T (FINDPARMDES (GET CNTXT 'PARENT)
                                                                    PARM))))))

(DEFUN GLMKATOM (NAME)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "edited: 11-Nov-82 11:54 -- borrowed from GLISP")
(IL:* IL:|;;;;| "Make a variable name for GLCOMP functions.")
(IL:* IL:|;;;;| "")

                                 (PROG (NEWATOM)
                                   LP  (SETQ NEWATOM (GENSYM (SYMBOL-NAME NAME)))
          (IL:* IL:|;;| "If an atom with this name has something on its proplist, try again.")

                                       (COND
                                          ((OR (GET NEWATOM 'RULE)
                                               (GET NEWATOM 'ISA))
                                           (GO LP))
                                          (T (RETURN NEWATOM)))))

(DEFUN MAKE-CONTEXT (CLASS PARENT)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Make a new data context of class CLASS with a pointer to PARENT context. Example: CLASS - CULTURE and PARENT = PATIEN
T1")
(IL:* IL:|;;;;| ".")

                                    (PROG (CNTXT)
                                          (SETQ CNTXT (GLMKATOM CLASS))
                                          (SETF (GET CNTXT 'ISA)
                                                CLASS)
                                          (COND
                                             (PARENT (SETF (GET CNTXT 'PARENT)
                                                           PARENT)))
                                          (RETURN CNTXT)))

(OEFUN PARMGET (CNTXT PARM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "Get a parameter value, backchaining for it if needed.")
```

```
(IL:* IL:|;;;;| "")

                                (OR (CNTXTGET CNTXT PARM)
                                    (AND (NOT (MEMBER PARM PREVGOALS))
                                         (BC-GOAL CNTXT PARM (CONS PARM PREVGOALS)))))

(DEFUN RUNRULE (CNTXT RULENAME)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Run a rule")
(IL:* IL:|;;;;| "")

                                (PROG (TALLY RULE RUNRULERULENAME)
                                      (SETQ RUNRULERULENAME RULENAME)
                                      (SETQ RULE (GET RULENAME 'RULE))
                                      (SETQ TALLY (EVAL (CADR RULE)))
                                      (COND
                                         ((AND TALLY (NUMBERP TALLY)
                                               (> TALLY 0.2))
                                          (RETURN (EVAL (CADDR RULE)))))))
```

(IL:* IL:|;;;;| "")

(IL:* IL:|;;;;| "EMYCIN-style comparisons")

```
(DEFUN GREATEQ* (X Y) (COND
                           ((AND X Y (NUMBERP X)
                                 (NUMBERP Y)
                                 (>= X Y))
                            1.0)))

(DEFUN GREATERP* (X Y) (COND
                            ((AND X Y (NUMBERP X)
                                  (NUMBERP Y)
                                  (> X Y))
                             1.0)))

(DEFUN LESSEQ* (X Y) (COND
                          ((AND X Y (NUMBERP X)
                                (NUMBERP Y)
                                (<= X Y))
                           1.0)))

(DEFUN LESSP* (X Y) (COND
                         ((AND X Y (NUMBERP X)
                               (NUMBERP Y)
                               (< X Y))
                          1.0)))

(DEFUN BETWEEN* (X LOW HIGH) (COND
                                  ((AND X LOW HIGH (NUMBERP X)
                                        (NUMBERP LOW)
                                        (NUMBERP HIGH)
                                        (<= LOW X)
                                        (< X HIGH))
                                   1.0)))
```

(IL:* IL:|;;;;| "")

(IL:* IL:|;;;;| " Printing functions,")

```
(DEFUN ENGLRULE (RULENAME)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;| " Uses CNTXTNAME, the last context given to defcontext.")
          (IL:* IL:|;;| " Prints out a rule in stylized English.")
          (IL:* IL:|;;| "")

                                   (PROG (RULE CONSE)
                                         (OR (SETQ RULE (GET RULENAME 'RULE))
                                             (RETURN NIL))
                                         (AM.NEW.LINE)
                                         (AM.NEW.LINE)
                                         (PRINC.STRING "If:")
                                         (AM.NEW.LINE)
                                         (PRINTPREMISES (CADR RULE)
                                                0 5 0)
                                         (AM.NEW.LINE)
                                         (AM.NEW.LINE)
                                         (PRINC.STRING "then:")
                                         (AM.NEW.LINE)
                                         (SETQ CONSE (CADDR RULE))
                                         (COND
                                            ((EQ (CAR CONSE)
                                                 'CONCLUDE)
                                             (PRINTCONC CONSE))
                                            (T (MAPC #'(LAMBDA (X)
                                                               (PRINTCONC X)) (CDR CONSE))))
                                         (AM.NEW.LINE)))


(DEFUN PRINTCLAUSE (CLAUSE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Print the clauses which make up a rule.")
(IL:* IL:|;;;;| "")

                                  (LET ((PRED (CAR CLAUSE)))
                                       (COND
                                          ((MEMBER PRED '(SAME NOTSAME SAME.GLOBAL))
                                           (COND
                                              ((LISTP (CADR CLAUSE))
                                               (PRINC.STRING "the answer to ")
                                               (PRINT.QUESTION (CDADR CLAUSE))
                                               (PRINC.STRING " is ")
                                               (PRINC.STRING (CADR (CADDR CLAUSE))))
                                              ((EQ PRED 'SAME.GLOBAL)
                                               (PRINC.STRING "the global value of ")
                                               (PRINC.STRING (CADR CLAUSE))
                                               (PRINC.STRING " is ")
                                               (PRINC.STRING (COND
                                                                ((LISTP (CADDR CLAUSE))
                                                                 (CADR (CADDR CLAUSE)))
                                                                (T (CADDR CLAUSE)))))
                                              ((EQ (CADDDR CLAUSE)
                                                   'YES)
                                               (PRINC.STRING "the ")
                                               (PRINC.STRING CNTXTNAME)
                                               (PRINC.STRING (CONO
                                                                ((EQ PREO 'SAME)
                                                                 " is ")
                                                                (T " is not ")))
                                               (PRINC.STRING (CLAUSEPARM CLAUSE)))
                                              (T (PRINC.STRING "the ")
                                                 (PRINC.STRING (CLAUSEPARM CLAUSE))
                                                 (PRINC.STRING " of the ")
                                                 (PRINC.STRING CNTXTNAME)
                                                 (PRINC.STRING (COND
                                                                  ((EQ PREO 'SAME)
                                                                   " is ")
                                                                  (T " is not ")))
                                                 (PRINC.STRING (CADOOR CLAUSE)))))))
```

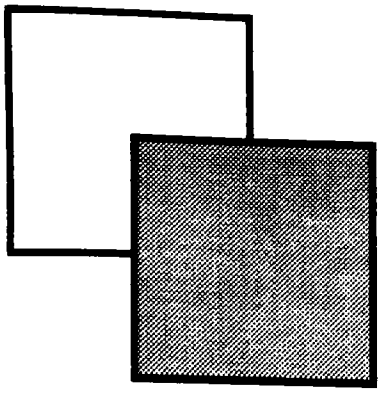
```
                                    ((MEMBER PRED '(GREATERP* GREATEQ* LESSP* LESSEQ* BETWEEN*))
                                     (PRINC.STRING "the ")
                                     (PRINC.STRING (CLAUSEPARM CLAUSE))
                                     (PRINC.STRING " of the ")
                                     (PRINC.STRING CNTXTNAME)
                                     (PRINC.STRING " is ")
                                     (PRINC.STRING (CADR (ASSOC PRED '((GREATERP* "greater than "
                                                                         )
                                                                       (GREATEQ*
                                                                      "greater than or equal to "
                                                                         )
                                                                       (LESSP* "less than")
                                                                       (LESSEQ*
                                                                          "less than or equal to"
                                                                         )
                                                                       (BETWEEN* "between ")))))
                                     (PRINC.STRING (CADDR CLAUSE))
                                     (COND
                                        ((EQ PRED 'BETWEEN*)
                                         (PRINC.STRING " and ")
                                         (PRINC.STRING (CADDDR CLAUSE)))))
                                    ((EQUAL PRED 'KNOWN)
                                     (PRINC.STRING "the ")
                                     (PRINC.STRING (CLAUSEPARM CLAUSE))
                                     (PRINC.STRING " of the ")
                                     (PRINC.STRING CNTXTNAME)
                                     (PRINC.STRING " is known")
                                     (COND
                                        ((CDDDR CLAUSE)
                                         (PRINC.STRING " to be ")
                                         (PRINC.STRING (CADODR CLAUSE))))))))

(DEFUN PRINTCONC (CONC)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;| "Prints out the conclusion in English.")
          (IL:* IL:|;;| "")

                              (PROG (TALLY)
                                    (COND
                                       ((NOT (EQ (CAR CONC)
                                                 'CONCLUDE))
                                        (RETURN NIL)))
                                    (SETQ TALLY (CADR (CDDOOR CONC)))
                                    (PRINC.STRING (COND
                                                     ((< (ABS TALLY)
                                                       500)
                                                      " there is weakly suggestive evidence ")
                                                     ((< (ABS TALLY)
                                                       800)
                                                      " there is suggestive evidence ")
                                                     ((< (ABS TALLY)
                                                       1000)
                                                      " there is strongly suggestive evidence ")
                                                     (T " it is definite ")))
                                    (PRINC.STRING "(")
                                    (PRINC.STRING (IL:FQUOTIENT (FLOAT TALLY)
                                                                1000.0))
                                    (PRINC.STRING ") that")
                                    (COND
                                       ((EQ (CADDDR CONC)
                                            'YES)
                                        (PRINC.STRING " the ")
                                        (PRINC.STRING CNTXTNAME)
                                        (PRINC.STRING (COND
                                                         ((< TALLY 0)
                                                          " is not ")
                                                         (T " is ")))
                                        (PRINC.STRING (CADDDR CONC)))
                                       (T (PRINC.STRING " the ")
                                          (PRINC.STRING (CADDR CONC))
                                          (PRINC.STRING " of the ")
                                          (PRINC.STRING CNTXTNAME)
                                          (PRINC.STRING (COND
```

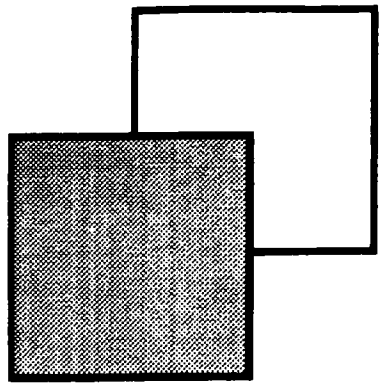
```
                                                    ((< TALLY 0)
                                                     " is not ")
                                                    (T " is ")))
                                        (PRINC.STRING (CADDDR CONC))))
                                (AM.NEW.LINE)))


(DEFUN PRINTCONCLUSION (CNTXT PARM)
          (IL:* IL:|;;| "")
          (IL:* IL:|;;| "Print conclusion for a parameter")
          (IL:* IL:|;;| "")

                                        (PROG NIL
                                              (AM.NEW.LINE)
                                              (PRINC.STRING PARM)
                                              (AM.NEW.LINE)
                                              (PRINT.STRING (CNTXTGET CNTXT PARM))
                                              (AM.NEW.LINE)))


(DEFUN PRINTPREMISES (CLAUSE NUMB SPACES INITSPACES)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Print premises of a rule")
(IL:* IL:|;;;;| "")

                                                          (PROG (JUNCTION)
                                                                (COND
                                                                   ((NULL CLAUSE)
                                                                    (RETURN NIL))
                                                                   ((MEMBER (CAR CLAUSE)
                                                                            '(SAME NOTSAME KNOWN
                                                                                   GREATERP* GREATEQ*
                                                                                   LESSP* LESSEQ*
                                                                                   BETWEEN* SAME.GLOBAL
                                                                                   ))
                                                                    (PRINTCLAUSE CLAUSE)
                                                                    (RETURN NIL)))
                                                                (SETQ JUNCTION (CAR CLAUSE))
                                                          LP    (COND
                                                                   ((NULL (SETQ CLAUSE (CDR CLAUSE)))
                                                                    (RETURN NIL)))
                                                                (CLSPACES (- SPACES INITSPACES))
                                                                (SETQ INITSPACES 0)
                                                                (SETQ NUMB (1+ NUMB))
                                                                (COND
                                                                   ((< NUMB 10)
                                                                    (PRINC.STRING " ")))
                                                                (PRINC.STRING NUMB)
                                                                (PRINC.STRING ") ")
                                                                (PRINTPREMISES (CAR CLAUSE)
                                                                       0
                                                                       (+ SPACES 5)
                                                                       9)
                                                                (COND
                                                                   ((CDR CLAUSE)
                                                                    (PRINC.STRING ", ")
                                                                    (PRINC.STRING
                                                                     (COND
                                                                        ((EQ JUNCTION '$AND)
                                                                         "and")
                                                                        ((EQ JUNCTION '$OR)
                                                                         "or")
                                                                        (T JUNCTION)))
                                                                    (AM.NEW.LINE)))
                                                                (GO LP)))


(DEFUN CLSPACES (N)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Print groups of spaces for neater printed format.")
(IL:* IL:|;;;;| "")

                                  (COND
                                     ((> N 0)
```

```
                              (PRINC.STRING " ")
                              (CLSPACES (1- N)))))


(DEFUN SHOWPROPS (ATM)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Show properties of a context, e.g., (SHOWPROPS 'ROCK3)")
(IL:* IL:|;;;;| "")

                             (PRINT.STRING (CONS ATM (SYMBOL-PLIST ATM))))


(DEFUN SHOWRULE (RULENAME)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " Show a rule in Lisp form e.g., (SHOWRULE 'RULE101)")
(IL:* IL:|;;;;| "")

                                (PRINT.STRING (GET RULENAME 'RULE)))



(IL:* IL:|;;;;| "")



(IL:* IL:|;;;;| " Questioning functions, Reasoning behind tool")


(DEFMACRO WHY (CNTXT PARM &OPTIONAL VALUE)·
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| "(WHY cntxt parm value) asks why a value was concluded. Value is optional. If unspecified, it defaults to the value concluded mos
t strongly.")
(IL:* IL:|;;;;| " Example: (WHY rock34 identity)")
(IL:* IL:|;;;;| "")

                                                  '(WHYEXPR ',CNTXT ',PARM ',VALUE))


(DEFUN WHYEXPR (CNTXT PARM VALUE)

(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " (WHY cntxt parm value) asks why a value was concluded. Value is optional; if unspecified, it defaults to the value concluded mo
st strongly. This function is called from the macro WHY.")
(IL:* IL:|;;;;| " Example: (WHY rock34 identity)")
(IL:* IL:|;;;;| "")

   (PROG (RULES RULENAME RULE TALLY CONCPART VALS)
         (COND
            ((NOT (GET CNTXT 'ISA))
             (PRINC.STRING CNTXT)
             (PRINC.STRING " ??")
             (RETURN NIL)))
         (OR VALUE (SETQ VALUE (VAL1EXPR CNTXT PARM)))
         (SETQ RULES (GET PARM 'RULES))
     LP  (COND
            ((NULL RULES)
             (RETURN NIL)))
         (SETQ RULENAME (CAR RULES))
         (SETQ RULE (GET RULENAME 'RULE))
         (SETQ RULES (CDR RULES))
          (IL:* IL:|;;| "See if this ruleconcluded the value of interest")

         (SETQ CONCPART (CADDR RULE))
         (COND
            ((EQ (CAR CONCPART)
                 'CONCLUDE)
             (SETQ VALS (LIST (LIST (CADDDR CONCPART)
                                    (CADR (CDDDDR CONCPART))))))
            ((EQ (CAR CONCPART)
                 'DO-ALL)
             (SETQ VALS (MAPCAN #'(LAMBDA (CONC)
                                          (CONO
```

```
                                          ((ANO (EQ (CAR CONC)
                                                    'CONCLUOE)
                                                (EQ (CAOOR CONC)
                                                    PARM))
                                           (LIST (LIST (CAOOOR CONC)
                                                       (CAOR (COOOOR CONC)))))) (COR CONCPART))))
      (T (AM.ERROR "Incorrect conclusion part of rule")))
   (CONO
      ((NOT (ASSOC VALUE VALS))
       (GO LP)))
    (IL:* IL:|;;| " Evaluate the precondition and see if it worked.")

   (SETQ TALLY (EVAL (CAOR RULE)))
   (CONO
      ((NOT (ANO TALLY (NUMBERP TALLY)))
       (GO LP)))
   (AM.NEW.LINE)
   (AM.NEW.LINE)
   (PRINC.STRING "The following rule concluded that ")
   (AM.NEW.LINE)
   (PRINC.STRING " the ")
   (PRINC.STRING PARM)
   (PRINC.STRING " of ")
   (PRINC.STRING CNTXT)
   (PRINC.STRING " was ")
   (PRINC.STRING VALUE)
   (PRINC.STRING " ( ")
   (PRINC.STRING (* TALLY (IL:FQUOTIENT (FLOAT (CAOR (ASSOC VALUE VALS)))
                                        1000.0)))
   (PRINC.STRING ") ")
   (AM.NEW.LINE)
   (AM.NEW.LINE)
   (ENGLRULE RULENAME)
   (GO LP)))


(OEFMACRO WHYNOT (CNTXT PARM VALUE)
(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " (WHYNOT cntxt parm value) asks why a value was not concluded.")
(IL:* IL:|;;;;| " Example (WHYNOT rock34 identity coal)")
(IL:* IL:|;;;;| "")

                                          '(WHYNOTEXPR ',CNTXT ',PARM ',VALUE))


(OEFUN WHYNOTEXPR (CNTXT PARM VALUE)

(IL:* IL:|;;;;| "")
(IL:* IL:|;;;;| " (WHYNOT cntxt parm value) asks why a value was not concluded.")
(IL:* IL:|;;;;| "This function is called by the macro WHYNOT")
(IL:* IL:|;;;;| " Example: (WHYNOT rock32 identity coal)")
(IL:* IL:|;;;;| "")

   (PROG (RULES RULENAME RULE TALLY CONCPART VALS ANTE)
      (CONO
         ((NOT (GET CNTXT 'ISA))
          (PRINC.STRING CNTXT)
          (PRINC.STRING " ??")
          (RETURN NIL)))
      (SETQ RULES (GET PARM 'RULES))
    LP (CONO
         ((NULL RULES)
          (RETURN NIL)))
      (SETQ RULENAME (CAR RULES))
      (SETQ RULE (GET RULENAME 'RULE))
      (SETQ RULES (COR RULES))
       (IL:* IL:|;;| "See if this rule concluded the value of interest.")

      (SETQ CONCPART (CAOOR RULE))
      (CONO
         ((EQ (CAR CONCPART)
              'CONCLUOE)
          (SETQ VALS (LIST (CAOOOR CONCPART))))
         ((EQ (CAR CONCPART)
```

```
                'DO-ALL)
            (SETQ VALS (MAPCAN #'(LAMBDA (CONC)
                                         (COND
                                            ((AND (EQ (CAR CONC)
                                                      'CONCLUDE)
                                                  (EQ (CADDR CONC)
                                                      PARM))
                                             (LIST (CADDDR CONC))))) (CDR CONCPART))))
              (T (AM.ERROR "Incorrect conclusion part of rule")))
           (COND
              ((NOT (MEMBER VALUE VALS))
               (GO LP)))
            (IL:* IL:|;;| " Evaluate the precondition and see if it worked.")

           (SETQ TALLY (EVAL (CADR RULE)))
           (COND
              ((AND TALLY (NUMBERP TALLY)
                    (> TALLY 0.2))
               (GO LP)))
           (AM.NEW.LINE)
           (AM.NEW.LINE)
           (PRINC.STRING "The following rule might have concluded that")
           (AM.NEW.LINE)
           (PRINC.STRING "the ")
           (PRINC.STRING PARM)
           (PRINC.STRING " of ")
           (PRINC.STRING CNTXT)
           (PRINC.STRING " was ")
           (PRINC.STRING VALUE)
           (AM.NEW.LINE)
           (AM.NEW.LINE)
           (SHOWRULE RULENAME)
           (AM.NEW.LINE)
           (PRINC.STRING " but it failed on the following condition:")
           (AM.NEW.LINE)
           (AM.NEW.LINE)
           (SETQ ANTE (CADR RULE))
           (COND
              ((NOT (EQ (CAR ANTE)
                        '$AND))
               (PRINT ANTE)
               (GO LP)))
        LPB (SETQ ANTE (CDR ANTE))
           (CONO
              ((NULL ANTE)
               (GO LP)))
           (SETQ TALLY (EVAL (CAR ANTE)))
           (COND
              ((AND TALLY (NUMBERP TALLY)
                    (> TALLY 0.2))
               (GO LPB)))
            (IL:* IL:|;;| "This clause failed. Print it and its result.")

           (PRINT.STRING (CAR ANTE))
           (PRINC.STRING " returned CF value was ")
           (PRINC.STRING TALLY)
           (AM.NEW.LINE)
           (GO LP)))

(IL:DECLARE\: IL:DONTCOPY
  (IL:FILEMAP (NIL)))
IL:STOP
```





For Foltman:henr801c

{DSK}<LISPFILES>THESIS>CODE>NEWRULES.;1

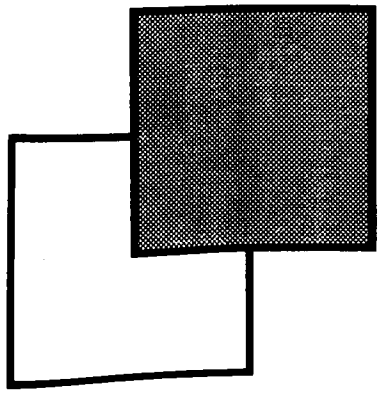
Created 23 – May – 89 14:54:59

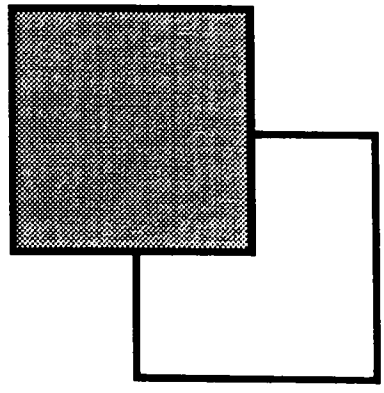
Printed 23 – May – 89 14:57:04

1 Sheet(s), 1 Copy.

Xerox Print Service 10.0 on Palette

Appearance Error (page 1): font problem; character [0B,36B] is not in font 'Terminal'. ... and 1 more





```
(DEFINE-FILE-INFO READTABLE "XCL" PACKAGE "INTERLISP")
(FILECREATED "20-Mar-89 16:38:25" {DSK}<LISPFILES>THESIS>CODE>NEWRULES.\;1 1590

      |previous| |date:| "17-Mar-89 13:50:18" {DSK}<LISPFILES>THESIS>CODE>NEWRULES.\;1)


(PRETTYCOMPRINT NEWRULESCOMS)

(RPAQQ NEWRULESCOMS ((VARS XCL-USER::NEW.RULES.LIST)))

(RPAQQ XCL-USER::NEW.RULES.LIST (XCL-USER::DEFRULES (RULE001 (XCL-USER::$AND (XCL-USER::SAME
                                                                                XCL-USER::CNTXT
                                                                                XCL-USER::CRITICAL
                                                                                XCL-USER::YES)
                                                                              (XCL-USER::SAME.GLOBAL
                                                                               XCL-USER::LINKER.USED
                                                                               'XCL-USER::LINK)
                                                                              (XCL-USER::SAME.GLOBAL
                                                                               XCL-USER::ERROR.TYPE
                                                                               'WARNING))
                                                                    (XCL-USER::CONCLUDE XCL-USER::CNTXT
                                                                           XCL-USER::SOLUTION BLAH
                                                                           XCL-USER::TALLY 1000.0))))
(DECLARE\: DONTCOPY
  (FILEMAP (NIL)))
STOP
```

AMUSED Index

A
ADA 36
AI 12
Antecedent 28, 30
Antecedent-driven 29
Assertion 12
ATN 7

B
Backtracking 19
Backward-chaining 14
Boyd 6
BOYD84 6
Break-and-examine 10
BringOver 21, 22
BTree 25

C
Cedar 20
Check-in 22
Check-out 22
CL 66
CML 66
CommonLisp 26
CONDE85 23, 24
CRON85 17

D
Database 5, 11, 12, 66
Dataflow 7, 9
Defcontext 29
Defrules 30
Dependencies 19
DF 3, 21, 22
DFD 7
DFTool 3, 19, 21, 22
Diagnosis 21, 28
Domain-dependent 13, 14

DSD 32

E
EMYCIN 28, 29, 30, 31
EPROMs 3
Ericsson 12, 13
Error-locating 9, 10, 11, 15
ES 19
Ethernet 23, 35

F
FILE86 21
FileTool 3
Fortran 27
FOUE84 15, 16
FTP 3, 22

G
Gosseyn 16

H
HARA83 6, 7, 9
HARA85 17
Harandi 6
HAYE83 29
HOWTO85 20

I
IL 66
IncludeChecker 23
Intel 2
Interlisp 26, 27,28, 66
Interlisp-D 32
Internet 35
Interpreter 19

K
KAU 17
KBPA 17
Knowledge-based 6, 12, 17
KORE86 9, 11
Koto 32, 66

L
LAMP83 20
LEWIS 1, 23
Link8051 3, 2, 65, 67
Lisp 3, 19, 26, 27, 30,

M
MacLisp 25, 26
MANU84 6
Manucci 6
MARC84 5
Medley 66
Mesa 1, 21, 22, 36, 69
Metaknowledge 16
MIF 2
Modeler 21
MYCIN 10, 28

N
NORD86 13, 15
Novak 29, 30

O
Oakland 9, 10
Object-level 12
Object-oriented 11, 12

P
Paradigm 6
PARC 20
POZZO 13, 14
Prototype 8, 10, 13, 14

Q
Quintus Prolog 31

S
Schemas 7, 9
SCHMIDT82 1
SL 19
SModel 20, 21, 23
Software-development 9
Spindles 15

STEELE84 25, 26
Sub-assemblies 5
Submodels 9
Symptom-fault 10

T
TANI87 28
TAU 17
Telecom 11, 12
TMYCIN 26, 27, 30, 37, 66

U
User-level 26

V
VerifyDF 20, 24

W
WATE86 28
WELI84 12

X
XCL 66
XDE 3, 68, 69
XLisp 19

Z
ZetaLisp 26, 27